LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
Stanford Research Park
700 Hansen Way
Palo Alto, CA 94304

Telephone
(650) 849-6600

Facsimile
(650) 849-6666

FACSIMILE TRANSMITTAL

TO		FROM	
Name:	Examiner DARROW, Justin T.	Name:	Andrew B. Schwaab
Loc:	Group 2132 / USPTO	Phone No.:	650-849-6643
Fax No.:	703-872-9306	Fax # Verified by:	abs
Phone No.:		# Pages (incl. this):	273
Subject:	US Appln. No. 09/764,370	Date:	August 4, 2004
Your File No.:	09/764,370	Our File No.:	07451.0001-17

Message:

In re Application of:

Karl L. Ginter, et al.

Application No.: 09/764,370

Filed: January 19, 2001

For: SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Group Art Unit: 2132

Examiner: Justin T. Darrow

PART 2 of 3

Examiner Darrow:

Here is the final summary of the related litigation, regarding which Microsoft has taken a comprehensive license to InterTrust's patent portfolio for a one-time payment of $440,000,000.00.

Please call if you have any questions or would like any additional information.

Sincerely,
Andrew Schwaab
direct 650-849-6643

If there is a problem with this transmission, notify fax room at (650) 849-6600 or the sender at the number above.

secure containers to other apparatuses or for the receipt of secure containers from other apparatuses; and	for transmitting or receiving secure containers. For example, RM-enabled OUTLOOK is designed to transmit and receive encrypted IRM-governed emails to/from other devices.
a second apparatus including:	
user controls, a communications port, a processor, a memory containing a second rule,	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM. The second rule governs use of an IRM protected document (e.g., an IRM rule permitting a document to be read by specified users or barring access to IRM-governed information from specified users, applications, or other principals).
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM-enabled device contains hardware or software for receiving and opening secure containers. The secure email has the capacity to contain an IRM-governed email item, with a rule being associated with each secure containers.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said apparatus, said protected processing environment including hardware or software used for applying said second rule and a secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item;	Protected information on the RM-enabled device is protected by the use of at least cryptographic technique. The secure container rule is an IRM rule governing access to the IRM protected document (e.g., a rule permitting editing by specified users). The rule governing the email works together with an additional rule to determine what access to or use (if any) are allowed with respect to the IRM-governed item (the document's content). For example, the additional rule may be received together with the rule in the use license, may be associated with a publishing license, may be associated with user certification, revocation lists, or exclusion policies, or may be received from any other source.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses; and	The device includes hardware or software used for transmitting or receiving secure containers. For example, RM-enabled OUTLOOK is designed to transmit and receive encrypted IRM-governed emails to/from other devices.
an electronic intermediary, said intermediary including a user rights authority clearinghouse.	The RMS Server (Microsoft hosted or otherwise) constructs a 'use license' specific to a piece content and targets it to a specific user.

29. A system as in claim 28, said user rights authority clearinghouse operatively connected to make rights available to users.	The RMS server sends *use licenses* to users through a communications port, e.g., Ethemet, serial, satellite, "the internet" These use licenses include rights. The clearing functionality of the RMS is operatively connected to the RMS server.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

28.	Product Infringing: Windows Media Rights Manager and Windows Media Player
A system including:	
(a) a first apparatus including;	Consumer's computer, as shown in WMRM SDK
(1) user controls,	Consumer's computer, as shown in WMRM SDK
(2) a communications port,	Consumer's computer, as shown in WMRM SDK
(3) a processor,	Consumer's computer, as shown in WMRM SDK
(4) a memory containing a first rule,	Memory is in the consumer's computer, first rule is a right received as part of a signed license (WMRM SDK, Step 9)
(5) hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	Consumer's computer receives Windows Media file (secure container) via communications port (WMRM SDK, Step 3) and applies secure container rule or rules via Windows Media Player and Windows Media Rights Manager.
(6) a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first rule and a secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item; and	Processing environment includes Windows Media Rights Manager and Windows processes for protecting operation of Windows Media Rights Manager
(7) hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses; and	Hardware or software employed in transmitting Windows Media files, including for example consumer's computer's communication port and Windows Media Player (WMRM SDK, Step 3)
(b) a second apparatus including:	2nd consumer's computer
(1) user controls,	2nd consumer's computer
(2) a communications port,	2nd consumer's computer
(3) a processor,	2nd consumer's computer
(4) a memory containing a second rule,	Memory is in the 2nd consumer's computer, first rule is a Right received as part of a signed license (WMRM SDK, Step 9)
(5) hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain	2nd consumer's computer receives Windows Media file (secure container) via communications port (WMRM SDK, Step 3) and applies secure container rule or rules via

a governed item, a secure container rule being associated with each of said secure containers;	Windows Media Player and Windows Media Rights Manager.
(6) a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said apparatus; said protected processing environment including hardware or software used for applying said second rule and a secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item;	Processing environment includes Windows Media Rights Manager and Windows processes for protecting operation of Windows Media Rights Manager; processing environment applies multiple rules in combination
(7) hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses; and	Hardware or software employed in transmitting Windows Media files, including for example 2nd consumer's computer's communication port and Windows Media Player. (WMRM SDK, Step 3)
(c) an electronic intermediary, said intermediary including a user rights authority clearinghouse.	License Issuer
29. A system as in claim 28,	
said user rights authority clearinghouse operatively connected to make rights available to users.	License Issuer, operatively connected to consumer's computer (WMRM SDK, Step 9)

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
56.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A method of securely delivering an item, including the following steps:	
performing an authentication step;	The RM-enabled application, e.g., Word, OUTLOOK, PowerPoint, etc., must be authenticated before it is allowed access to or use of the content.
associating a digital signature with said item;	The RM protected content is signed.
incorporating said item into a first secure electronic container, said item being at least in part encrypted while in said container, said incorporation occurring in an apparatus containing a first protected processing environment, said protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said apparatus;	RM-protected content is packaged with rules and encrypted. Protected information on the RM enabled computer is protected by the use of at least cryptographic techniques.
in said protected processing environment, associating a first rule with said first secure electronic container, said first rule at least in part governing at least one aspect of access to or use of said item;	The IRM-protected document (said item) has an associated rule or rules.
authenticating an intended recipient of said item;	A recipient of IRM-protected content must be authenticated before being allowed access to or use of the content.
transmitting said first secure electronic container and said first rule to said intended recipient; and	The document is sent via IRM-protected email as an attachment.
using a second protected processing environment, providing said intended recipient access to at least a portion of said item, said access being governed at least in part by said first rule and by a second rule present at said intended recipient's site.	The email is received at another IRM-enabled computer. The first said rule is the rule(s) associated with the attached document, and the second rule is the rule(s) received that govern the email itself.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

126.	Product Infringing: Windows Hardware Quality Labs Authentication services, Windows operating Systems (such as Windows XP) that support the driver signing features, and any product using Driver Signing feature
A method of providing trusted intermediary services including the following steps:	
at a first apparatus, receiving an item from a second apparatus;	Microsoft's Window Hardware Quality Labs (WHQL) (first apparatus) receiving driver package (item) from independent hardware vendor (IHV) or any driver developer (second apparatus).
associating authentication information with said item;	The signature information of a security catalog file (see next element of claim) names Microsoft as the publisher. WHQL's signature is intended to signify that a driver has complied with Microsoft's Windows compatibility and/or Secure Audio Path (SAP) specifications.
incorporating said item into a secure digital container;	The hashes of the files making up the driver package are included in the signed security catalog file for the driver package. The catalog file makes the driver package a secure digital container.
associating a first rule with said secure digital container, said first rule at least in part governing at least one aspect of access to or use of said item;	Driver developers specify rules in an INF file that govern the installation and/or use of the driver. For example, as specified in the INF, the installation events will vary based on the user's operating system version, which includes architecture, product type and suite. The INF logging rules and can further specify security rules that are evaluated when the driver is used. White Paper – Operating-System Versioning for Drivers under Windows XP Setup selects the [*Models*] section to use based on the following rules: If the INF contains [*Models*] sections for several major or minor operating system version numbers, Setup uses the section with the highest version numbers that are not higher than the operating system version on which the installation is taking place.

If the INF [*Models*] sections that match the operating system version also include product type decorations, product suite decorations, or both, then Setup selects the section that most closely matches the running operating system.

Suppose, for example, Setup is running on Windows XP Professional (which is operating system version 5.1), and it finds the following entry in a [Manufacturer] section:

%FooCorp%=FooMfg, NT, NT.5, NT.5.5, NT....0x80

In this case, Setup will look for a [*Models*] section named [FooMfg.NT.5]. Setup will also use the [FooMfg.NT.5] section if it is running on a Datacenter version of Windows .NET Server, because a specific major/minor version takes precedence over the product type and suite mask.

For example, to create an INF that is intended for use only on Windows XP, the INF file could contain the following:

```
[Manufacturer]
"Foo Corp." = FooMfg, NT.5.1, NT.5.2
[FooMfg.NT.5.1]
"Foo Device" = FooDev, *FOO1234
```

Note the omission of the undecorated [FooMfg] section, as well as the omission of the [FooMfg.NT.5.2] section. This INF file would appear to be "empty" on any operating system other than Windows XP.

<u>**Access Control List Rules**</u>

<u>**XP DDK – Tightening File-Open Security in a Device INF File**</u>
For Microsoft Windows 2000 and later, Microsoft tightened file-open security in the class installer INFs for certain device classes, including CDROM, DiskDrive, FDC, FloppyDisk, HDC, and SCSIAdapter.
If you are unsure whether the class installer for your device has tightened security on file opens, you should tighten security by using the device's INF file to assign a value to the **DeviceCharacteristics** value name in the registry. Do this within an *add-*

	registry-section, which is specified using the INF AddReg directive.
transmitting said secure digital container and said first rule to a third apparatus, said third apparatus including a protected processing environment at least in part protecting information stored in said protected processing environment from tampering by a user of said third apparatus;	Microsoft, IHV, driver developer or any other party distributing signed driver packages transmitting the driver package to user (third apparatus). Since the driver package includes the INF file, it will include the first rule. The protected processing environment (PPE) is Windows operating system with its pertinent services such as Windows File Protection, signature and cryptographic functions, Plug and Play and Set-up and their related default and modifiable policies. The PPE checks for signatures on driver packages and detects situations when the driver package's signature does not match the driver package. Additionally, the Digital Rights Manager (DRM) components (kernel and client) will contribute to making the third apparatus a PPE when the SAP functionality is invoked. [That is, when SAP is required, an additional signature is checked to verify that the driver is SAP compliant and that it hasn't been tampered with.]
said third apparatus receiving said secure digital container and said first rule;	The end-user receiving the driver package.
said third apparatus checking said authentication information; and	A step in the Plug and Play/Setup driver installation process checks signature at installation. Additionally, the DRM component will check the DRM signature when invoking DRM functionality. <u>White Paper – Driver Signing for Windows</u> During driver installation, Windows compares the hashes contained in the driver's CAT file with the computed hash of the driver binaries to determine whether the binaries have changed since the CAT file was created. If a driver fails the signature check or there is no CAT file, what happens next depends on the driver signing policy in effect on the user's system: If the policy is set to Ignore, the driver installs silently, with no message to the user. If the policy is set to Warn, a message warns the user the driver is unsigned, which means that it has not passed WHQL

	testing and might cause problems. The Warn dialog box gives an administrative user the option to override the warning and install an unsigned driver anyway. If the policy is set to Block, the system displays a message that informs the user that the driver cannot be installed because it is not digitally signed.
said third apparatus performing at least one action on said item, said at least one action being governed, at least in part, by said first rule and by a second rule resident at said third apparatus prior to said receipt of said secure digital container and said first rule, said action governance occurring at least in part in said protected processing environment.	The action would be installing and/or using the driver. For example, installation policies govern the actions (ignore, warn or block) taken based on whether a driver is signed or not and these policies (rule) are resident on the third apparatus. Another rule is the "ranking" of available drivers when selecting a driver to install. This ranking process includes whether a driver is signed or not. Another rule is the security access rules that the class installer that will be used to install the device has. In the case of DRM, the content will have associated rules governing its use in a SAP-complaint environment. These rules (the content license) can be resident at the third apparatus particularly in the case when a user is installing a new (SAP-compliant) device that will render previously acquired content or in the case that acquired content cannot be rendered until the user installs required drivers. For example, when installing: The XP driver ranking process and the modifiable default related to signature state of the driver act as the second rule. The driver will be installed only if the first and second rules validate. <u>Operating-System Versioning for Drivers under Windows XP</u> *Default System Policy for Unsigned Drivers* If the user installs an unsigned driver for a designated device class from disk or from another web site, Windows XP/Windows 2000 displays a warning that the driver is unsigned, thus helping to preserve the integrity of the released system. However, by default, Windows XP/Windows 2000

	does not block installation of unsigned drivers, so vendors can get urgent hot-fixes to customers while waiting for WHQL to test the fix. In Windows XP, the default driver signing policy can be changed through the Hardware tab of the System applet on the Control Panel. A user can change the policy to be more restrictive, but not less restrictive on a per-user basis (that is, a user can change Warn to Block, but not to Ignore). An administrator can change the policy to be either more restrictive or less restrictive for all users on the system by checking "Apply the setting as system default." *Driver Ranking* Under Windows XP, the driver ranking strategy has been modified as follows: If an INF file is unsigned, and if neither the [*Models*] section nor the [*DDInstall*] section is decorated with an NT-specific extension, the INF file is considered "suspect" and its rank is shifted into a higher range (that is, worse) than all hardware and compatible rank matches of INF files for which one (or both) of those criteria are met. The new ranking ranges will now be: 0 – 0xFFF (DRIVER_HARDWAREID_RANK) : "trusted" hardware-ID match 0x1000 – 0x3FFF : "trusted" compatible-ID match 0x8000 – 0x8FFF : "untrusted" hardware-ID match 0x9000 – 0xBFFF : "untrusted" compatible-ID match 0xC000 - 0xCFFF : "untrusted" undecorated hardware-ID match (possibly a Windows 9x-only driver) 0xD000 - 0xFFFF : "untrusted" undecorated compatible-ID match (possibly a Windows 9x-only driver)
127. A method as in claim 126, in which said authentication information at least in part identifies said first apparatus and/or a	The authentication information will identify Microsoft, operator of the first apparatus.

user of said first apparatus.	

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

126.	Products Infringing: Microsoft Software that includes the Authenticode feature, .NET Framework SDK, Visual Studio, Microsoft technology that supports a digital signature function (such as ActiveX), Windows Installer technology.
A method of providing trusted intermediary services including the following steps:	Infringement is based on use Microsoft ActiveX control, Cabinet file, Microsoft Windows Installer, Authenticode and Software Restriction Policy technologies. For example, a software publisher distributing a signed application that has licensed ActiveX controls embedded within it would practice this method.
at a first apparatus, receiving an item from a second apparatus;	The item is unsigned software such as an ActiveX control or any software packaged in a cabinet file or Microsoft Installer (.msi) file. Within the development environment, multiple software developers (working on a second apparatus) will send their unsigned software to a secure location (first apparatus) containing the entity's private signing key. An example entity would be a software publisher. Source: Deploying ActiveX Controls on the Web with the Internet Component Download **The holder of the digital certificate** Keeping your digital certificate safe is very important. Some firms (including Microsoft) do not keep their signature file on site. The signature is kept with the Certificate Authority and files are sent there for signing.
associating authentication information with said item;	Signing the software associates the software publisher's identify with the software. Source: Packaging ActiveX Controls Signing Cabinet Files A .cab file can be digitally signed like an ActiveX control. A digital signature provides accountability for software developers: The signature associates a software vendor's name with a given file. A signature is applied to a .cab file (or control) using the Microsoft Authenticode®

	technology. The .cab tool set assists software developers in applying digital signatures to .cab files by allowing a developer to allocate space in the .cab file for the signature.
incorporating said item into a secure digital container;	Signing software either directly or within a package (cabinet or .msi file) secures it in a digital container. Alternately, the signed ActiveX control could be placed into a signed cabinet file.
associating a first rule with said secure digital container, said first rule at least in part governing at least one aspect of access to or use of said item;	The first rule would be the licensing support code within the ActiveX control and/or conditional syntax statements when the software is within a signed .msi file. When the software is within a signed cabinet file, the first rule can be a rule contained in the software, as is the case when an ActiveX control is packaged in a signed cabinet file. First rule, in the case of ActiveX: When an application with a licensed ActiveX control is started, an instance of the control usually needs to be created. The application accomplishes this by making a call to CreateInstanceLic and passing the license key embedded in the application as a parameter in the call. The ActiveX control performs a string comparison between the embedded license key and its own copy of the license key. If the keys match, an instance of the control is created and the application can execute normally. Source: Using ActiveX Controls to Automate Your Web Pages Run-time licensing Most ActiveX Controls should support design-time licensing and run-time licensing. (The exception is the control that is distributed free of charge.) Design-time licensing ensures that a developer is building his or her application or Web page with a legally purchased control; run-time licensing ensures that a user is running an application or displaying a Web page that contains a legally purchased control. Design-time licensing is verified by control containers such as Visual Basic, Microsoft Access, or Microsoft Visual InterDev®. Before these containers allow a developer to place a control on a form or Web page,

	they first verify that the control is licensed by the developer or content creator. These containers verify that a control is licensed by calling certain functions in the control: If the license is verified, the developer can add it. Run-time licensing is also an issue for these containers (which are sometimes bundled as part of the final application); the containers again call functions in the control to validate the license that was embedded at design time.
transmitting said secure digital container and said first rule to a third apparatus, said third apparatus including a protected processing environment at least in part protecting information stored in said protected processing environment from tampering by a user of said third apparatus;	The third apparatus is a user computer or an application server. The protected processing environment (PPE) is Windows operating system, Internet Explorer (IE) and pertinent operating IE services such as Windows File Protection and security, signature and cryptographic functions related to code signing and related policies. The PPE checks for signatures on software or the software packages and detects situations when the signature does not validate as an indication that tampering may have occurred with the item.
said third apparatus receiving said secure digital container and said first rule;	Having the third apparatus receiving said secure digital container and said first rule is typical of networked computing environments.
said third apparatus checking said authentication information; and	Examine the signature information includes verifying that signature was creating using the private key that corresponds to the public key of the publisher.
said third apparatus performing at least one action on said item, said at least one action being governed, at least in part, by said first rule and by a second rule resident at said third apparatus prior to said receipt of said secure digital container and said first rule, said action governance occurring at least in part in said protected processing environment.	The action would be installation and/or use of the distributed software. The second rule can be software restriction policies resident on the machine, which can be invoked at installation and/or runtime. .NET Framework Security – pg 259 and White Paper – Using Software Restriction Policies in Windows XP and Windows .NET Server to Protect Against Unauthorized Software Software Restriction Polices is a policy-driven technology that allows administrators to set code-identity-based rules that determine whether an application is allowed to execute. (.NET Framework Security – pg 259)

	For example, administrators can set rules for all Windows Installer packages coming from the Internet or Intranet zone. As part of the DLL load mechanisms, Software Restriction Policies is invoked and starts to check its most specific rules. Software Restriction Policies get invoked prior to an .exe being able to run. The four types of rules are – hash, certificate, path, and zone. Note: The hash and certificate rules relate directing to the signature information whereas, the path and zone rules do not.
127. A method as in claim 126, in which said authentication information at least in part identifies said first apparatus and/or a user of said first apparatus.	The software publisher, user of first device, is identified in the authentication information.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

126.	Product infringing: Visual Studio .NET, .NET Framework SDK, Authenticode, Products that contain the .NET CLR, Compact CLR or CLI.
A method of providing trusted intermediary services including the following steps:	
at a first apparatus, receiving an item from a second apparatus;	First apparatus is a software build or deployment services computer that has access to signing key. The item may be a program, graphic, media object or other resource, from a developer computer, or archive (second apparatus).
associating authentication information with said item;	Associating a cryptographic hash with the file that will contain this item for the purpose of ensuring the authenticity of the item, along with names and attributes that are desired to be associated with the item for identification purposes.
incorporating said item into a secure digital container;	Producing signed, strongly named assembly that contains this assembly and associated attributes.
associating a first rule with said secure digital container, said first rule at least in part governing at least one aspect of access to or use of said item;	Including any security demands (such as members of the Microsoft .NET Framework SDK Public Class CodeAccessSecurityAttribute) as part of the assembly.
transmitting said secure digital container and said first rule to a third apparatus, said third apparatus including a protected processing environment at least in part protecting information stored in said protected processing environment from tampering by a user of said third apparatus;	The third apparatus is a user computer or an application server. The third apparatus's protected processing environment is Windows NT and the .NET CLR, CLI and/or compact CLR. Information is protected from tampering because user is not administrator, user runs code on server, a share on another computer, or over a network. Further this information is protected by a number of protection mechanisms that are included with the Windows NT and CLR, CLI and/or compact CLR distributions.
said third apparatus receiving said secure digital container and said first rule;	Having the third apparatus receiving said secure digital container and said first rule is typical of networked computing environments.
said third apparatus checking said authentication information; and	The .NET Framework, when the assembly is installed into the global assembly cache (GAC), verifies the strong name of assemblies. This process includes verifying that signature was creating using the private key that corresponds to the

293482.02

	public key of the publisher.
said third apparatus performing at least one action on said item, said at least one action being governed, at least in part, by said first rule and by a second rule resident at said third apparatus prior to said receipt of said secure digital container and said first rule, said action governance occurring at least in part in said protected processing environment.	The action is executing code that is the item or using code that renders the item. Action is governed by security demands on code that calls the item or on code that calls code included in the .NET assembly that manages said item. The second rule is the machine, enterprise, user, and application configuration file resident rules. Typically these configuration files will be populated before the arrival of most new assemblies in a virtual distribution environment. This action governance occurs in the protected processing environment of the CLR, CLI and/or compact CLR.
127. A method as in claim 126, in which said authentication information at least in part identifies said first apparatus and/or a user of said first apparatus.	The authentication information will identify the .NET Assembly Class company name and trademark attributes that identify the apparatus or user of the first apparatus as being a member of an entity or a branded source (brand name).

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,185,683

126.	Product infringing: Visual Studio .NET, .NET Framework SDK, Authenticode, Products that contain the .NET CLR, Compact CLR or CLI.
A method of providing trusted intermediary services including the following steps:	
at a first apparatus, receiving an item from a second apparatus;	The item is an unsigned .NET assembly, which can include, but not be limited to, a Web control, multi-file assembly or component. Within the development environment, multiple assembly builders (working on a second apparatus) will send their unsigned assembly to a secure location (first apparatus) containing the entity's private signing key. An example entity would be a software publisher. .NET Security Framework – pg 130-1 Describes this exact practice and further explains the "Delay Signing Assemblies" feature of .NET that accommodates the fact that "many publishers will keep the private key in a secure location, possibly embedded in specially designed cryptographic hardware." "Delay signing is a technique used by developers whereby the public key is added to the assembly name as before, granting the assembly its unique identity, but no signature is computed. Thus, no private key access is necessary."
associating authentication information with said item;	Strong naming the assembly binds the entity's/publisher's name into the assembly. The public portion of the key used to strongly name the assembly is placed in the assembly manifest. Other assemblies or applications can contain references to the strong names of strongly named assemblies such as in the case of applications that contain references to a set of compliant .NET core libraries. Strong naming compliant .NET core libraries with the European Computers Manufactures Association's (ECMA) key is a way to allow any publisher to develop compliant .NET core libraries that can be authenticated by other applications.

	<u>.NET Security Framework – pg 124</u> "Strong naming is a process whereby an assembly name can be further qualified by the identity of the publisher." <u>.NET Security Framework – pg 133</u> The publisher must advertise its public key or keys in an out-of-band fashion (such as documentation shipped with the product or on the company Web site) <u>.NET Security Framework – pg 130</u> The goal of the ECMA key is to allow a slightly more generalized strong name binding than usual, namely allowing binding to the publisher of the runtime in use, rather than to a fixed publisher.
incorporating said item into a secure digital container;	Signing the assembly places it in a secure container. <u>.NET Framework Security – pg 527</u> Strong named assemblies cannot be modified in any manner without destroying the strong name signature. <u>Applied Microsoft .NET Framework Programming – pg 89</u> *Strongly Named Assemblies Are Tamper-Resistant* When the assembly is installed into the GAC, the system hashes the contents of the file containing the manifest and compares the hash value with the RSA digital signature value embedded within the PE file (after unsigning it with the public key). If the values are identical, the file's contents haven't been tampered with and you know that you have the public key that corresponds to the publisher's private key. In addition, the system hashes the contents of the assembly's other files and compares the hash values with the hash values stored in the manifest file's FileDef table. If any of the hash values don't match, at least one of the assembly's files has been tampered with and the assembly will fail to install into the GAC.
associating a first rule with said secure digital container, said first rule at least in part governing at least one aspect of access to or use of said item;	A .NET assembly includes imperative and declarative statements/rules that will govern its access or use. For example, role-based security or strong name demands in the assembly can be the first rule. MSDN on Role-Based Security Applications that implement role-based security grant rights based on the role

	associated with a principal object. The principal object represents the security context under which code is running. The PrincipalPermission object represents the identity and role that a particular principal class must have to run. To implement the PrincipalPermission class imperatively, create a new instance of the class and initialize it with the name and role that you want users to have to access your code. MSDN on StrongNameIdentityPermission StrongNameIdentityPermission class defines the identity permission for strong names. StrongNameIdentityPermission uses this class to confirm that calling code is in a particular strong-named assembly.
transmitting said secure digital container and said first rule to a third apparatus, said third apparatus including a protected processing environment at least in part protecting information stored in said protected processing environment from tampering by a user of said third apparatus;	The third apparatus is a user computer or an application server. The software publisher transmitting the .NET assembly to an end-user with a CLR. The third apparatus's protected processing environment is Windows NT and the .NET CLR, CLI and/or compact CLR. Information is protected from tampering because user is not administrator, user runs code on server, a share on another computer, or over a network. Further this information is protected by a number of protection mechanisms that are included with the Windows NT and CLR, CLI and/or compact CLR distributions.
said third apparatus receiving said secure digital container and said first rule;	The end-user receiving the signed assembly.
said third apparatus checking said authentication information; and	The .NET Framework, when the assembly is installed into the global assembly cash (GAC), verifies the strong name of assemblies. This process includes verifying that signature was creating using the private key that corresponds to the public key of the publisher. Applied Microsoft .NET Framework Programming – pg 89 *Strongly Named Assemblies Are Tamper-Resistant* As above. .NET Framework Security – pg 128 The verification of any strong name assemblies is performed automatically when needed by the .NET Framework. Any assembly claiming a strong name but

	failing verification will fail to install into the global assembly or download cache or will fail to load at runtime.
said third apparatus performing at least one action on said item, said at least one action being governed, at least in part, by said first rule and by a second rule resident at said third apparatus prior to said receipt of said secure digital container and said first rule, said action governance occurring at least in part in said protected processing environment.	Within the CLR (protected processing environment), the execution of the program will depend upon whether the user is of the "role" required of the assembly or whether the calling assembly is from a strong-named assembly specified in the "item" assembly (alternate first rules) and only if assembly complies with the local code access security policy (second rule), as an example of one of the types of rules that .NET Framework allows to be resident on the third apparatus..
127. A method as in claim 126, in which said authentication information at least in part identifies said first apparatus and/or a user of said first apparatus.	The user of the first apparatus is the developer at the assembly developer. Strong naming binds the publisher's name to assembly.

LaMacchia, Brian, etc, .NET Framework Security, Addison-Wesley, 2002
Richter, Jeffrey, Applied Microsoft .NET Framework Programming, Microsoft Press, 2002

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,253,193

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
1	Infringing products include Windows Media Player and Windows Media Rights Manager SDK
A method comprising:	
(a) receiving a digital file including music;	Reference is made to the Windows Media Rights Manager SDK Programming Reference ("WMRM SDK"), attached hereto as Exhibit A. Media Player infringement analysis is set forth herein using the example of a music file downloaded and transferred to a portable audio player. Consumer receives a Windows Media file (WMRM SDK, Step 3)
(b) storing said digital file in a first secure memory of a first device;	Windows Media file is stored in consumer's computer and all use of it is securely managed by the Secure Content Manager in Windows Media Player.
(c) storing information associated with said digital file in a secure database stored on said first device, said information including at least one budget control and at least one copy control, said at least one budget control including a budget specifying the number of copies which can be made of said digital file; and said at least one copy control controlling the copies made of said digital file;	License is stored in the License Store (WMRM SDK, Step 5); license includes Rights which may include AllowTransfertoNonSDMI, AllowTransfertoSDMI, (or Allow Transfer to WM-D-DRM-Compliant devices or other types of devices), and TransferCount- the number of times a piece of content may be transferred to the device (a transfer budget).
(d) determining whether said digital file may be copied and stored on a second device based on at least said copy control;	Windows Media Rights Manager enforces the license restrictions
(e) if said copy control allows at least a portion of said digital file to be copied and stored on a second device,	Windows Media Rights Manager determines whether the AllowTransferToNonSDMI or AllowTransferToSDMI rights are present.(Or, Allow Transfer to WM-D-DRM-Compliant devices or other types of devices.)
(1)copying at least a portion of said digital file;	Transfer to the SDMI or non-SDMI portable device (Allow Transfer to WM-D-DRM-Compliant devices or other types of devices), if allowed by Windows Media Rights Manager
(2)transferring at least a portion of said digital file to a second device including a memory and an audio and/or video output;	Portable device necessarily includes at least a memory and audio output
(3)storing said digital file in said memory of said second device; and	Music file is transferred to the portable device
(4)including playing said music through said audio output.	Portable device plays the music
2. A method as in claim 1, further comprising:	
(a) at a time substantially contemporaneous with said transferring step, recording in said	Counter reflecting TransferCount is decremented by Windows Media Rights

first device information indicating that said transfer has occurred.	Manager
3. A method as in claim 2, in which:	
(a) said information indicating that said transfer has occurred includes an encumbrance on said budget.	Counter decrement reduces the allowable number of budgeted transfers
4. A method as in claim 3, in which:	
(a) said encumbrance operates to reduce the number of copies of said digital file authorized by said budget.	Counter decrement reduces the allowable number of budgeted transfers

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,253,193

	Infringing products include Windows Media Player and Windows Media Rights Manager SDK
11. A method comprising:	
(a) receiving a digital file;	Consumer receives a Windows Media file (WMRM SDK, Step 3)
(b) storing said digital file in a first secure memory of a first device;	Windows Media file is stored in consumer's computer and all use of it is securely managed by the Secure Content Manager in Windows Media Player.
(c) storing information associated with said digital file in a secure database stored on said first device, said information including a first control;	License information is stored in the License Store (WMRM SDK, Step 10), license information includes Rights. License Rights may include AllowTransferToNonSDMI, AllowTransferToSDMI (Allow Transfer to WM-D-DRM-Compliant devices or other types of devices), TransferCount
(d) determining whether said digital file may be copied and stored on a second device based on said first control,	WMRM determines whether transfer rights are included in license (WMRM SDK, Step 5)
(1) said determining step including identifying said second device and determining whether said first control allows transfer of said copied file to said second device, said determination based at least in part on the features present at the device to which said copied file is to be transferred;	Portable Device Service Provider Module identifies the portable device as either SDMI-compliant or non-SDMI-compliant (or WM-D-DRM Compliant or other types of supported devices) and provides this information to Windows Media Device Manager, which allows the transfer based on whether the device identification matches the License Right.
(e) if said first control allows at least a portion of said digital file to be copied and stored on a second device,	If Windows Media Rights Manager determines whether the AllowTransferToNonSDMI or AllowTransferToSDMI rights are present (or Allow Transfer to WM-D-DRM-Compliant devices or other types of devices), the following steps are performed:
(1) copying at least a portion of said digital file;	Transfer to the SDMI or non-SDMI (Allow Transfer to WM-D-DRM-Compliant or other) portable device, if allowed by Windows Media Rights Manager
(2) transferring at least a portion of said digital file to a second device including a memory and an audio and/or video output;	Portable device necessarily includes at least a memory and audio output
(3) storing said digital file in said memory of said second device; and	Music file is stored in the portable device
(4) rendering said digital file through said output.	Portable device plays the music

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,253,193

	Product infringing: Windows Media Player, Windows Media Player, Windows Media Rights Manager SDK
15. A method comprising:	
(a) receiving a digital file;	Consumer receives a Windows Media file ((WMRM SDK, Step 3)
(b) an authentication step comprising:	
(1) accessing at least one identifier associated with a first device or with a user of said first device; and	License includes identity of user's Windows Media Player: WM Players capable of playing protected content must be individualized. They contain a unique (Individualized) DRM client component to which protected WMA content licenses are bound. Content licenses are bound to this DRM individualization module as the result of a challenge sent from the Client to the WMLM service. The challenge contains information about Individualized DRM Client (in the form of an encrypted Client ID) and capabilities of the machine (e.g. support for Secure Audio Path (SAP), version of the WMRM SDK supported in the player.
(2) determining whether said identifier is associated with a device and/or user authorized to store said digital file;	Music file cannot be used unless identifier indicated in License matches user's Windows Media Player identifier (that is, the Individualized DRM Client to which the license is bound must be the same one supported by the device).
(c) storing said digital file in a first secure memory of said first device, but only if said device and/or user is so authorized, but not proceeding with said storing if said device and/or user is not authorized;	Music file will not be processed through Windows Media Player, including protected rendering buffers, unless the identifiers match. Protected WMA file can be stored on client even if unauthorized but it cannot be decrypted and enter into the secure boundary (first secure memory) of the player unless appropriately licensed.
(d) storing information associated with said digital file in a secure database stored on said first device, said information including at least one control;	License includes Rights and is stored in the License Store, Rights may include AllowTransferToNonSDMI, AllowTransferToSDMI, (or Allow Transfer To WM-D-DRM-CompliantDevice or other device) TransferCount
(e) determining whether said digital file may be copied and stored on a second device based on said at least one control;	Windows Media Rights Manager enforces the license restrictions
(f) if said at least one control allows at least a portion of said digital file to be copied and stored on a second device,	If appropriate rights are present, the following steps are performed:
(1) copying at least a portion of said	Transfer to the SDMI or non-SDMI (or WM-

digital file;	D-DRM Compliant or other) portable device, if allowed by Windows Media Rights Manager
(2) transferring at least a portion of said digital file to a second device including a memory and an audio and/or video output;	Portable device necessarily includes at least a memory and audio output
(3) storing said digital file in said memory of said second device; and	Music file is stored in the portable device
(4) rendering said digital file through said output.	Portable device plays the music
16. A method as in claim 15, in which:	
said digital file is received in an encrypted form; and further comprising: decrypting said digital file after said authentication step and before said step of storing said digital file in said memory of said first device.	Protected Windows Media File is encrypted. WMP will not decrypt file until license is processed. Licenses are bound to Individualization DLLs, which are bound to Hardware ID. Ind. DLL and Hardware ID must be verified as the Ids to which the license is bound – this is the authentication process. (Recall that this module was created based in part on receipt of the Client Hardware ID or fingerprint and the license was create based in part on receipt of a challenge from the client indicating the security properties (SAP-ready, SDK support, etc.) of the client).

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,253,193

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
19.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A method comprising:	
receiving a digital file at a first device;	Receiving a digital file such as a Word Document, email, Excel spreadsheet, PowerPoint presentation, or other content at a recipient's device. Such content may be received via email, received on removable media, such as floppy disk, downloaded and viewable by Internet Explorer, *e.g.*, a web page possibly containing graphics and/or audio data, etc.
establishing communication between said first device and a clearinghouse located at a location remote from said first device;	If the digital file is subject to rights management, and the recipient tries to open the digital file in an IRM-enabled application, the IRM-enabled application contacts a remote RMS, *i.e.*, clearinghouse for a use license.
said first device obtaining authorization information including a key from said clearinghouse;	If the recipient is authorized to access or use the digital file, the RMS creates a license for the digital file. The RMS then seals a key inside the license so that only the recipient canaccess or use the digital file. Finally, the RMS sends the license back to the recipient.
said first device using said authorization information to gain access to or make at least one use of said first digital file, including using said key to decrypt at least a portion of said first digital file; and	The recipient's device then uses the key in the license to gain access or decrypt a portion of the digital file.
receiving a first control from said clearinghouse at said first device;	The license received from the RMS at the recipient's device contains at least one control, such as restricting the ability to print, forward, or edit.
storing said first digital file in a memory of said first device;	The digital file is stored in the memory of the said recipient's device, such as in RAM, on a hard drive, etc.
using said first control to determine whether said first digital file may be copied and stored on a second device;	The at least one control in the license limits copying the digital file. Such controls are set when the digital file was authored. For example, when the digital file is authored, the IRM-enabled application presented the author with a list of policy templates with different rights levels. The author selected an appropriate rights level which may for instance, allow other users in the system to open and read the document, but not

	to modify it, copy text from it, or forward it. These rights or controls are then associated with the digital file. When an attempt is made to access the digital file, the RMS determines the recipient's rights based on the recipient's identity and the policies or controls associated with the digital file.
if said first control allows at least a portion of said first digital file to be copied and stored on a second device,	If the control in the license allows copying the digital file to a second device, then at least a portion of the digital file is copied,
copying at least a portion of said first digital file;	such as by transferring or forwarding the digital file in an email message;
transferring at least a portion of said first digital file to a second device including a memory and an audio and/or video output;	A portion of the digital file is then transferred to a second device, such as a personal computer or portable device. The second device includes a memory and an audio and/or video output. The memory may be a hard-drive, RAM, CD, DVD, or other storage. The audio and/or video output may be speakers and/or a video monitor.
storing said first digital file portion in said memory of said second device; and	The digital file is stored in the second device's memory.
rendering said first digital file portion through said output.	The digital file is rendered through the output, such as played through the speakers and/or displayed on the video monitor. For example, a Word document is displayed on the screen of the video monitor.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.

INTERTRUST INFRINGEMENT CHART

FOR U.S. PATENT NO. 6,253,193

	Infringing products include Windows Media Player, Windows Media Rights Manager SDK
19. A method comprising:	
(a) receiving a digital file at a first device;	WMRM SDK, Step 3.
(b) establishing communication between said first device and a clearinghouse located at a location remote from said first device;	WMRM SDK, Step 6.
(c) said first device obtaining authorization information including a key from said clearinghouse;	WMRM SDK, Step 9. [License contains the key]
(d) said first device using said authorization information to gain access to or make at least one use of said first digital file, including using said key to decrypt at least a portion of said first digital file; and	WMRM SDK, Step 11.
(e) receiving a first control from said clearinghouse at said first device;	WMRM SDK, Steps 8-9.
(f) storing said first digital file in a memory of said first device;	WMRM SDK, Step 3.
(g) using said first control to determine whether said first digital file may be copied and stored on a second device;	At least the following WMRMRights Object properties meet this limitation: AllowTransferToNonSDMI, AllowTransferToSDMI (or AllowTransfer To WM-D-DRM-Compliant Device or other) and TransferCount
(h) if said first control allows at least a portion of said first digital file to be copied and stored on a second device,	This and all subsequent claim steps occur when the condition specified in the WMRMRights Object property is met
(i) copying at least a portion of said first digital file;	Transfer to the SDMI or non-SDMI (or WM-D-DRM Compliant) portable device, if allowed by Windows Media Rights Manager
(j) transferring at least a portion of said first digital file to a second device including a memory and an audio and/or video output;	Portable device necessarily includes at least a memory and audio output
(k) storing said first digital file portion in said memory of said second device; and	Music file is stored in the portable device
(l) rendering said first digital file portion through said output.	Portable device plays the music

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,253,193

	Infringing products include Windows Media Player, Windows Media Player, Windows Media Rights Manager SDK
51. A method comprising:	
(a) receiving a digital file at a first device;	WMRM SDK, Step 3.
(b) establishing communication between said first device and a clearinghouse located at a location remote from said first device;	WMRM SDK, Step 6.
(c) said first device obtaining authorization information from said clearinghouse; and	WMRM SDK, Step 9.
(d) said first device using said authorization information to gain access to or make at least one use of said first digital file;	WMRM SDK, Step 11.
(e) storing said first digital file in a memory of said first device;	WMA file stored on client
(f) using at least a first control to determine whether said first digital file may be copied and stored on a second device, said determination based at least in part on (1) identification information regarding said second device, and (2) the functional attributes of said second device;	If device is based on WM D-DRM, it has a certificate that is used to identify the device as compliant as well as the device's security level. The security level indicates support on the device for such attributes as an internal clock.
(g) if, based at least in part on said identification information, said first control allows at least a portion of said first digital file to be copied and stored on a second device,	If License specifies that transfer of protected WMA file to WM-D-DRM-Compliant device is allowed, transfer may occur.
(h) copying at least a portion of said first digital file;	If transfer is a licensed right as indicated in the license, the song is copied to the device via Windows Media Device Manager.
(i) transferring at least a portion of said first digital file to a second device including a memory and an audio and/or video output;	Windows Media Device Manager transfers the content to the device:
(j) storing said first digital file portion in said memory of said second device; and	WMA file is stored on device
(k) rendering said first digital file portion through said output.	WMA file is rendered.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
33.	Infringing products include all Microsoft tools that support the Microsoft ActiveX licensing model, Visual Studio .NET, the Microsoft Installer SDK, and Operating System products that include the Microsoft Installer technology.
A data processing arrangement comprising at least one storing arrangement that at least temporarily stores a first secure container comprising first protected data and a first set of rules governing use of said first protected data,	The first protected data is an ActiveX control. The first alternative for the first secure container is the signed .msi in which the ActiveX developer packaged the ActiveX control. The first set of rules is the conditional syntax statements of the signed .msi file. The second alternative for the first secure container is the signed and licensed ActiveX control. The first set of rules is the license support code in the ActiveX control. A third alternative for the first container is a signed cabinet file containing a (signed or unsigned) ActiveX control with license support code. The first set of rules is the license support code in the ActiveX control.
and at least temporarily stores a second secure container comprising second protected data different from said first protected data and a second set of rules governing use of said second protected data; and	The second protected data is the application developer's application that includes/uses the ActiveX control. The application developer's signed .msi file (second secure container) contains the application (second protected data). The second set of rules is the signed .msi file's conditional syntax statements that will be governed the offer/installation of the application.
a data transfer arrangement, coupled to at least one storing arrangement, for transferring at least a portion of said first protected data and a third set of rules governing use of said portion of said first protected data to said second secure container.	Placing the licensed ActiveX control (first protected information) in a signed cabinet file (third secure container) that itself is included in the application's signed .msi file (second secure container). The third set of rules is the license support code in the ActiveX control.
further comprising	
means for creating and storing, in said at least one storing arrangement, a third secure container;	The ability of the application developer to package files in signed cabinet files.

283482.02

said data transfer arrangement further comprising means for transferring said portion of said first protected data and said third set of rules to said third secure container, and means for incorporating said third secure container within said second secure container.	The third secure container is a cabinet file signed by the application developer and including at least the licensed ActiveX control (first protected information. The licensing support code in the ActiveX control when its developer added licensing support to the ActiveX control is the third set of rules.
34. A data processing arrangement as in claim 33 further comprising means for applying said third set of rules to govern at least one aspect of use of said portion of said first protected data.	Before an ActiveX control will create a copy of itself, the calling application has to pass a license key to the ActiveX control. The license support code in the ActiveX control (third rule set) evaluates the authenticity of the calling application's request.
35. A data processing arrangement as in claim 34 further comprising means for applying said second set of rules to govern at least one aspect of use of said portion of said first protected data.	Windows Installer operating system service enforces the conditional syntax statements of the application's signed .msi file. These statements govern the offer/installation of the ActiveX control.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

41	Infringing products include all Microsoft tools that support the Microsoft ActiveX licensing model, Visual Studio .NET, the Microsoft Installer SDK, and Operating System products that include the Microsoft Installer technology.
A method comprising performing the following steps within a virtual distribution environment comprising one or more electronic appliances and a first secure container, said first secure container comprising (a) a first control set, and	The signed .msi file created by the ActiveX control developer is the first secure container. The conditional syntax statement(s) of the ActiveX control developer's signed .msi file is/are the first control set.
(b) a second secure container comprising a second control set and first protected information:	The first protected information is the ActiveX control. The first alternative for the second secure container is the signed and licensed ActiveX control. The second control set is the license support code in the ActiveX control. The second alternative for the second secure container is a signed cabinet file containing the (signed or unsigned) ActiveX control. The second control set is the license support code in the ActiveX control.
using at least one control from said first control set or said second control set to govern at least one aspect of use of said first protected information while said first protected information is contained within said first secure container;	The ActiveX control developer's conditional syntax statements (first control set) in the ActiveX developer's signed .msi file govern the offer/installation of the ActiveX control while it is in its signed .msi file. Alternately, the license support code (second control set) in the ActiveX control governs use of the licensed ActiveX control.
creating a third secure container comprising a third control set for governing at least one aspect of use of protected information contained within said third secure container;	The third secure container is a signed .msi file. The application developer packages its application in a signed .msi file (third secure container) and includes conditional syntax statements (third control set) in the signed .msi
incorporating a first portion of said first protected information in said third secure container, said first portion made up of some or all of said first protected information; and	Placing the ActiveX control into the application developer's signed .msi file (third secure container).
using at least one control to govern at least	The application developer's conditional

293462.02

one aspect of use of said first portion of said first protected information while said first portion is contained within said third secure container.	syntax statement(s) in its signed .msi file govern the offer/installation ActiveX control while it is in the signed .msi file (third secure container).

42. A method as in claim 41, in which said first secure container further includes a fourth secure container comprising a fourth control set and second protected information and further comprising the following step:	The second protected information is a second ActiveX control. The first alternative for the fourth secure container is the signed and licensed second ActiveX control. The fourth control set is the license support code in the ActiveX control. The second alternative for the fourth secure container is a signed cabinet file containing the (signed or unsigned) second ActiveX control. The fourth control set is the license support code in the ActiveX control.
using at least one control from said first control set or said fourth control set to govern at least one aspect of use of said second protected information while said second protected information is contained within said first secure container.	The ActiveX control developer's conditional syntax statements (first control set) in the ActiveX developer's signed .msi file govern the offer/installation of the second ActiveX control while it is in its signed .msi file. Alternately, the license support code (second control set) in the ActiveX control governs use of the licensed ActiveX control.

47. A method as in claim 41, in which said step of creating a third secure container includes:	
creating said third control set by incorporating at least one control not found in said first control set or said second control set.	The application developer's conditional syntax statements are not found in either the first control set or the second control set.

52. A method as in claim 41 in which said step of creating a third secure container occurs at a first site, and further comprising:	
copying or transferring said third secure container from said first site to a second site located remotely from said first site.	The application developer at first site distributes its application to other sites.

53. A method as in claim 52 in which said first site is associated with a content distributor.	The application developer at the first site is the content distributor.

54. A method as in claim 53 in which said second site is associated with a user of	The application developer distributes the application to end-users.

content.	
55. A method as in claim 54 further comprising the following step:	
said user directly or indirectly initiating communication with said first site.	For Internet downloads, the user initiates the communication with the first site.
64. A method as in claim 54 in which said third control set includes one or more controls at least in part governing the use by said user of at least a portion of said first portion of said first protected information.	The application developer's conditional syntax statements (third control set) govern the installation of the ActiveX control (first protected information).
76. A method as in claim 41 in which said creation of said third secure container further comprises using a template which specifies one or more of the controls contained in said third control set.	The third secure container is the application developer's signed .msi file and the third control set is the conditional syntax statements in that file. Microsoft supplies several template .msi databases for use in authoring installation packages. The UISample.msi is the template recommended in the "An Installation Example" on MSDN. This template msi files contains several default conditional syntax statements. At least two of these conditional syntax statements directly govern the installation by blocking progress until the EULA is accepted.
78. A method as in claim 52 in which said creation of said third secure container further comprises using a template which specifies one or more of the controls contained in said third control set.	The third secure container is the application developer's signed .msi file and the third control set is the conditional syntax statements in that file. Microsoft supplies several template .msi databases for use in authoring installation packages. The UISample.msi is the template recommended in the "An Installation Example" on MSDN. This template msi files contains several default conditional syntax statements. At least two of these conditional syntax statements directly govern the installation by blocking progress until the EULA is accepted.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

81.	Infringing products include all Microsoft tools that support the Microsoft ActiveX licensing model, Visual Studio .NET, the Microsoft Installer SDK, and Operating System products that include the Microsoft Installer technology.
A data processing arrangement comprising:	
a first secure container comprising first protected information and a first rule set governing use of said first protected information;	The first alternative for the first secure container is the ActiveX control developer's signed .msi file containing a licensed ActiveX control (the first protected information). The conditional syntax statements of the signed .msi file are the first rule set. The second alternative for the first secure container is the signed cabinet file containing the ActiveX control. The license support code in the ActiveX control is the first rule set. The third alternative for the first secure container is the licensed and signed ActiveX control governed by license support code in the ActiveX control.
a second secure container comprising a second rule set;	The second secure container is the signed .msi file which the application developer package its application. The second rule set is the conditional syntax statements of the application developer's signed .msi file.
means for creating and storing a third secure container; and	The third container is a signed cabinet file containing at least the ActiveX control.
means for copying or transferring at least a portion of said first protected information and a third rule set governing use of said portion of said first protected information to said second secure container, said means for copying or transferring comprising:	Putting the licensed ActiveX control (first protected information) in a signed cabinet file (third secure container). The licensing support code in the ActiveX control is third rule set.
means for incorporating said third secure container within said second secure container.	Packaging the signed cabinet file in the signed .msi file.

82. A data processing arrangement as in claim 81 further comprising:	
means for applying at least one rule from said third rule set to at least in part govern at least one factor related to use of said portion of said first protected information.	The third rule set ensures the user is licensed.

83. A data processing arrangement as in claim 82 further comprising:	

293482.02

means for applying at least one rule from said second rule set to at least in part govern at least one factor related to use of said portion of said first protected information.	The second rule set governs the offer/installation of first protected information.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

85.	Infringing products include all Microsoft tools that support the Microsoft ActiveX licensing model, Visual Studio .NET, the Microsoft Installer SDK, and Operating System products that include the Microsoft Installer technology.
A method comprising the following steps:	
creating a first secure container comprising a first rule set and first protected information;	The first protected information is the ActiveX control. The first alternative for the first secure container is the signed and licensed ActiveX control. The first rule set is the license support code in the ActiveX control. The second alternative for the first secure container is an (signed or unsigned) ActiveX control with license support contained within a signed cabinet file. The first rule set is the ActiveX license support code.
storing said first secure container in a first memory;	The first secure container is stored at the ActiveX control developer's location.
creating a second secure container comprising a second rule set;	The second secure container is the application developer's signed .msi file. The conditional syntax statements of the signed .msi file are the second rule set.
storing said second secure container in a second memory;	The second secure container is stored at the application developer's location.
copying or transferring at least a first portion of said first protected information to said second secure container, said copying or transferring step comprising:	The ActiveX control developer packages the control in a signed .msi file for distribution to the application developer's site.
creating a third secure container comprising a third rule set;	The third secure container is the ActiveX control developer's signed .msi file containing a licensed ActiveX control. The conditional syntax statements of the signed .msi file are the third rule set.
copying said first portion of said first protected information;	In preparation for using a msi authoring tool, such as Microsoft's Orca, copying the ActiveX control to a package staging area.
transferring said copied first portion of said first protected information to said third secure container; and	Using msi authoring tool to import the control into the signed .msi file.
copying or transferring said copied first portion of said first protected information from said third secure container to said second secure container.	The application developer installs the ActiveX control, which involves removing it from the ActiveX developer's signed .msi file and installing it into its environment. Subsequently, the

	application developer places the ActiveX control into its signed .msi file when it is packaging its application.
87. A method as in claim 85 in which said copied first portion of said first protected information consists of the entirety of said first protected information.	The entire ActiveX control is copied.
89. A method as in claim 85 in which	
said first memory is located at a first site,	The first memory is located at the ActiveX control developer's site.
said second memory is located at a second site remote from said first site, and	The second memory is located at the application developer's site.
said step of copying or transferring said first portion of said first protected information to said second secure container further comprises copying or transferring said third secure container from said first site to said second site.	The ActiveX control developer's signed .msi file is transferred from its site to the site of the application developer.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

85. (alternate infringing scenario).	Infringing products include all Microsoft tools that support the Microsoft ActiveX licensing model, Visual Studio .NET, the Microsoft Installer SDK, and Operating System products that include the Microsoft Installer technology.
A method comprising the following steps:	
creating a first secure container comprising a first rule set and first protected information;	The first protected information is the ActiveX control. The first alternative for the first secure container is the signed and licensed ActiveX control. The first rule set is the license support code in the ActiveX control. The second alternative for the first secure container is a (signed or unsigned) ActiveX control with license support contained within a signed cabinet file. The first rule set would remain the ActiveX license support code. The third alternative for the first secure container is a signed msi file in which the ActiveX control developer packaged its ActiveX control. The first rule set is the conditional syntax statement(s) of the signed msi file.
storing said first secure container in a first memory;	The first secure container is stored at the ActiveX control developer's location.
creating a second secure container comprising a second rule set;	The second secure container is the application developer's signed .msi file. The conditional syntax statements of the signed .msi file are the second rule set.
storing said second secure container in a second memory;	The second secure container is stored at the application developer's location.
copying or transferring at least a first portion of said first protected information to said second secure container, said copying or transferring step comprising:	The ActiveX control is placed in a cabinet file signed by the application developer and the signed cabinet file is placed in a .msi file signed by the application developer.
creating a third secure container comprising a third rule set;	The third secure container is signed cabinet file in which the application developer placed licensed ActiveX. The third rule set is the license support code in the ActiveX control.
copying said first portion of said first protected information;	Copying ActiveX control.
transferring said copied first portion of said first protected information to	Transferring ActiveX control to signed cabinet file.

293482.02

Exhibit B
110

said third secure container; and	
copying or transferring said copied first portion of said first protected information from said third secure container to said second secure container.	The application developer places the signed cabinet file into its signed .msi file when it is packaging its application.
87. A method as in claim 85 in which said copied first portion of said first protected information consists of the entirety of said first protected information.	The entire ActiveX control is copied.
93. A method as in claim 85 in which	
said step of copying transferring said copied first portion of said first protected information from said third secure container to said second secure container further comprises storing said third secure container in said second secure container.	The ActiveX control is placed in a cabinet file signed by the application developer and the signed cabinet file is placed in a .msi file signed by the application developer.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

1:	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
A method of operating on a first secure container arrangement having a first set of controls associated therewith, said first secure container arrangement at least in part comprising a first protected content file, said method comprising the following steps performed within a virtual distribution environment including at least one electronic appliance:	The first protected content is a signed and licensed .NET component used by the .NET assembly. The .NET assembly is distributed with a signed and governed .msi file.. The second protected content is another signed and licensed .NET component that is used by the .NET assembly.
using at least one control associated with said first secure container arrangement for governing, at least in part, at least one aspect of use of said first protected content file while said first protected content file is contained in said first secure container arrangement;	The first protected content is signed and licensed .NET component (first secure container) contained within the .NET assembly. The one control is a declarative statement(s) within the assembly's header.
creating a second secure container arrangement having a second set of controls associated therewith, said second set of controls governing, at least in part, at least one aspect of use of any protected content file contained within said second secure container arrangement;	The protected content is the same as the first protected content plus the additional implementation information included in the signed .msi file. The second secure container is the signed .msi file created for the .NET assembly. The signed .msi file's conditional syntax statements are the second set of controls that control the offer/installation of the .NET assembly.
transferring at least a portion of said first protected content file to said second secure container arrangement, said portion made up of at least some of said first protected content file; and	The entire .NET assembly is included in the signed .msi file. Packaging the .NET assembly in the signed .msi file involves the following process steps. In preparation for using a msi authoring tool, such as Microsoft's Orca, copying the .NET component to a package staging area. Using msi authoring tool to import the .NET component into the signed .msi file.
using at least one rule to govern at least one aspect of use of said first protected content file portion while said portion is contained within said second secure container arrangement:	The conditional syntax statement(s) of the signed .msi file (second secure container) control(s) the offer/installation of the .NET assembly.
in which	
said first secure container arrangement comprises a third secure container	The first alternative for the third secure container is a licensed and signed .NET

arrangement comprising a third set of controls and said first protected content file, and	component governed by the set of declarative statements comprising the LicenseProviderAttribute (third set of controls). The second alternative for the third secure container is a .NET component whose hash is included in the header of the .NET assembly. The set of declarative statements comprising the LicenseProviderAttribute is the third set of controls.
said first secure container arrangement further comprises a fourth secure container arrangement comprising a fourth set of controls and a second protected content file.	The first alternative for the fourth secure container is another licensed and signed .NET component governed by the set of declarative statements comprising the LicenseProviderAttribute (fourth set of controls). The second alternative for the fourth secure container is the container created when the hash of the .NET component is included in the header information of the .NET assembly. The set of declarative statements comprising the LicenseProviderAttribute is the fourth set of controls.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

33.	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
A data processing arrangement comprising at least one storing arrangement that at least temporarily stores a first secure container comprising first protected data and a first set of rules governing use of said first protected data,	The first protected information is the .NET component. The first alternate for the first secure container is the signed .msi file in which the .NET component developer packaged its .NET component. The first set of rules is the conditional syntax statements of the signed .msi file. The second alternative for the first secure container is a licensed and signed .NET component governed by the set of declarative statements comprising the LicenseProviderAttribute of the .NET component (first set of controls). The third alternative for the first container is a signed cabinet file containing a (signed or unsigned) .NET component with license support. The first set of controls is the set of declarative statements comprising the LicenseProviderAttribute of the .NET component.
and at least temporarily stores a second secure container comprising second protected data different from said first protected data and a second set of rules governing use of said second protected data; and	The second protected data is the .NET assembly developer's assembly that includes/uses the .NET component. The first alternative for the second secure container is a signed .msi file in which the .NET assembly developer packaged its multi-file assembly (second protected data). The second set of rules is the conditional syntax statements of the signed .msi file that governs the offer/installation of the .NET assembly. The second alternative for the second secure container is a signed .NET assembly. The second set of rules is the declarative rules within the assembly's header.
a data transfer arrangement, coupled to at least one storing arrangement, for	The third secure container is a signed .NET assembly governed by declarative rules in

transferring at least a portion of said first protected data and a third set of rules governing use of said portion of said first protected data to said second secure container,	its header (third set of rules). An alternative third rule set is the set of declarative statements comprising the LicenseProviderAttribute. The .NET assembly includes the .NET component. The secure .NET assembly is included in a signed .msi file (second secure container). An alternative third secure container is the container created by hashing the .NET component and including the hash in the header information of a .NET assembly. The .NET component is included in the signed and governed .NET assembly (second secure container). The third set of rules is the set of declarative statements comprising the LicenseProviderAttribute. An alternative third secure container is a signed cabinet file containing the .NET component and which is destined for a signed .msi file (second secure container). The third set of rules is the set of declarative statements comprising the LicenseProviderAttribute.
further comprising	
means for creating and storing, in said at least one storing arrangement, a third secure container;	The first alternative for the third secure container is a signed .NET assembly. In this case, the second secure container is the signed .msi file. The second alternative for the third container is the container created by including a hash of the .NET component in the header information of a .NET assembly. In this case, the second secure container is either the signed .msi file or the signed .NET assembly. The third alternative for the third container is a cabinet file signed by the .NET assembly developer containing the .NET assembly and/or the .NET component. In this case the signed .msi file is the second secure container.
said data transfer arrangement further comprising means for transferring said portion of said first protected data and said third set of rules to said third secure container, and means for incorporating said third secure container within said second secure container.	The first alternative for the third secure container is the signed .NET assembly, which includes and/or uses the licensed .NET component (first protected information). The third set of rules is a declarative rule within the .NET assembly's header. The .NET assembly is placed in a signed .msi file (second secure container).

	The second alternative for the third secure container is the container that results when the hash of the .NET component is added to the .NET assembly header information. The third set of rules is the set of declarative statements comprising the LicenseProviderAttribute added to the assembly. The third alternative for the third secure container is a cabinet file signed by the .NET assembly developer containing the .NET assembly and/or the .NET component. The third set of rules is a declarative rule(s) within the .NET assembly's header and/or the set of declarative statements comprising the LicenseProviderAttribute added to the assembly
34. A data processing arrangement as in claim 33 further comprising means for applying said third set of rules to govern at least one aspect of use of said portion of said first protected data.	When the third rule set is the declarative statement(s) of the assembly header, the runtime CLR enforces the statements. When the third set of rules is the set of declarative statements comprising the LicenseProviderAttribute added to the assembly, the license support code in the .NET component evaluates the authenticity of the calling assembly's request.
35. A data processing arrangement as in claim 34 further comprising means for applying said second set of rules to govern at least one aspect of use of said portion of said first protected data.	When the second set of rules is the conditional syntax statements of the signed .msi file, the Windows Installer operating system service enforces the conditional syntax statements of .NET assembly's signed .msi file, which govern the offer/installation of the .NET component. When the second set of rules is the declarative statement(s) within the assembly's header, the runtime CLR enforces the statements.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

41.	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
A method comprising performing the following steps within a virtual distribution environment comprising one or more electronic appliances and a first secure container, said first secure container comprising (a) a first control set, and	The signed .msi file created by the .NET component developer is the first secure container. The first conditional syntax statement(s) of the .NET component developer's signed .msi file is/are the first control set.
(b) a second secure container comprising a second control set and first protected information:	The first protected information is the .NET component. The first alternative for the second secure container is the signed and licensed .NET component. The second control set is the set of declarative statements comprising the LicenseProviderAttribute. The second alternative for the second secure container is a signed cabinet file. The second control set remains the set of declarative statements comprising the LicenseProviderAttribute.
using at least one control from said first control set or said second control set to govern at least one aspect of use of said first protected information while said first protected information is contained within said first secure container;	The .NET component developer's conditional syntax statements (first control set) in its signed .msi file governs the offer/installation of the .NET component while it is in the signed .msi file. Alternately, the set of declarative statements comprising the LicenseProviderAttribute (second control set) of the licensed .NET component governs use of the .NET component.
creating a third secure container comprising a third control set for governing at least one aspect of use of protected information contained within said third secure container;	The first alternative for the third secure container is a signed .NET assembly, the protected information is the .NET component and the third control set is the declarative statement(s) within the .NET assembly's header. The second alternative for the third secure container is a signed .msi file in which the .NET assembly developer packages its .NET assembly and the third control set is the conditional syntax statement(s) in the signed .msi file.

incorporating a first portion of said first protected information in said third secure container, said first portion made up of some or all of said first protected information; and	In the first alternative, placing the .NET component into the signed .NET assembly. In the second alternative, placing the .NET component into the. Net assembly developer's signed msi file.
using at least one control to govern at least one aspect of use of said first portion of said first protected information while said first portion is contained within said third secure container.	In the first alternative, the .NET assembly developer's declarative statement(s) within the .NET assembly's header govern(s) the use of the .NET component while it is in the signed .NET assembly. In the second alternative, the conditional syntax statements of the .NET assembly developer's signed .msi file govern the offer/installation of the .NET component while it is in the signed .msi file.

42. A method as in claim 41, in which said first secure container further includes a fourth secure container comprising a fourth control set and second protected information and further comprising the following step:	The second protected information is a second .NET component. The first alternative for the fourth secure container is the signed and licensed second .NET component. The fourth control set is the set of declarative statements comprising the LicenseProviderAttribute of the second .NET component. The second alternative for the fourth secure container is a second signed cabinet file. The fourth control set is the set of declarative statements comprising the LicenseProviderAttribute.
using at least one control from said first control set or said fourth control set to govern at least one aspect of use of said second protected information while said second protected information is contained within said first secure container.	The .NET component developer's conditional syntax statements (first control set) in its signed .msi file governs the offer/installation of the second .NET component while it is in the signed .msi file. Alternately, the set of declarative statements comprising the LicenseProviderAttribute (fourth control set) of the licensed second .NET component governs use of the second .NET component.

47. A method as in claim 41, in which said step of creating a third secure container includes:	
creating said third control set by incorporating at least one control not found in said first control set or said second control set.	The .NET assembly developer's declarative statements (first alternative for third control set) and/or the developer's conditional syntax statements (second alternative for the third control set) are not found in either

293482.02

	the first control set or the second control set.
52. A method as in claim 41 in which said step of creating a third secure container occurs at a first site, and further comprising:	
copying or transferring said third secure container from said first site to a second site located remotely from said first site.	The .NET assembly developer at first site distributes its assembly to other sites.
53. A method as in claim 52 in which said first site is associated with a content distributor.	The .NET assembly developer's business module is used to create and distribute its assembly.
54. A method as in claim 53 in which said second site is associated with a user of content.	The .NET assembly developer distributes the assembly to end-users.
55. A method as in claim 54 further comprising the following step:	
said user directly or indirectly initiating communication with said first site.	For Internet downloads, the user initiates the communication with the first site.
64. A method as in claim 54 in which said third control set includes one or more controls at least in part governing the use by said user of at least a portion of said first portion of said first protected information.	When the third control set is the .NET assembly developer's declarative statement(s) within the .NET assembly's header, it governs the user's use of the .NET component (first protected information). When the third control set is the .NET assembly developer's conditional syntax statements of the .NET assembly developer's signed .msi file, it governs the user's offer acceptance/installation of the .NET component (first protected information).
76. A method as in claim 41 in which said creation of said third secure container further comprises using a template which specifies one or more of the controls contained in said third control set.	When the third secure container is the .NET assembly developer's signed .msi file and the third control set is the conditional syntax statements in that file. Microsoft supplies several template .msi databases for use in authoring installation packages. The UISample.msi is the template recommended in the "An Installation Example" on MSDN. This template msi files contains several default conditional syntax statements. At least two of these conditional syntax statements directly govern the installation by blocking progress until the EULA is accepted.

78. A method as in claim 52 in which said creation of said third secure container further comprises using a template which specifies one or more of the controls contained in said third control set.	When the third secure container is the .NET assembly developer's signed .msi file and the third control set is the conditional syntax statements in that file. Microsoft supplies several template .msi databases for use in authoring installation packages. The UISample.msi is the template recommended in the "An Installation Example" on MSDN. This template msi files contains several default conditional syntax statements. At least two of these conditional syntax statements directly govern the installation by blocking progress until the EULA is accepted.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

81.	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
A data processing arrangement comprising:	
a first secure container comprising first protected information and a first rule set governing use of said first protected information;	The first protected information is the .NET component. The first alternative for the first secure container is the signed .msi file in which the .NET component developer packaged its assembly. The first rule set is the conditional syntax statements written by the .NET component developer and placed into the signed .msi file. The second alternative for the first secure container is the signed cabinet file containing the (signed or unsigned) .NET component. The set of declarative statements comprising the LicenseProviderAttribute when its developer added licensing support to the assembly is the first rule set. The third alternative for the first secure container is the licensed and signed .NET component governed by the set of declarative statements comprising the LicenseProviderAttribute (first rule set) added by the .NET component developer.
a second secure container comprising a second rule set;	The first alternative for the second secure container is the signed .msi file in which the .NET assembly developer packaged its .NET assembly. The second rule set is the conditional syntax statements written by the .NET assembly developer and placed into the signed .msi file. The second alternative for the second secure container is the signed .NET assembly. The second rule set is the declarative statements in the .NET assembly's header.
means for creating and storing a third secure container; and	When the second secure container is the signed msi file, the third secure container is the signed .NET assembly. When the second secure container is the

Exhibit B

293482.02

	signed .NET assembly, the third secure container a .NET component secured by placing it in a signed cabinet file or by including its hash in the header of the assembly.
means for copying or transferring at least a portion of said first protected information and a third rule set governing use of said portion of said first protected information to said second secure container, said means for copying or transferring comprising:	When the second secure container is the signed msi file and the third secure container is the signed .NET assembly, the third rule set is the set of declarative statements within the assembly's header. When the second secure container is the signed .NET assembly, the third rule set is the set of declarative statements comprising the LicenseProviderAttribute (third rule set) added to the .NET component by its developer.
means for incorporating said third secure container within said second secure container.	When the second secure container is the signed msi file and the third secure container is the signed .NET assembly, the assembly is placed in the signed .msi file. When the second secure container is the signed .NET assembly and the third secure container is a .NET component contained in a signed cabinet file or a .NET component whose hash is included in the header of the assembly, the third secure container is incorporated within the .NET assembly.

82. A data processing arrangement as in claim 81 further comprising:	
means for applying at least one rule from said third rule set to at least in part govern at least one factor related to use of said portion of said first protected information.	When the third rule set is declarative statements within the assembly's header, it governs the use of the .NET assembly which includes the first protected information. When the third rule set is the set of declarative statements comprising the LicenseProviderAttribute added by the .NET component by its developer, it ensures the user is licensed.

83. A data processing arrangement as in claim 82 further comprising:	
means for applying at least one rule from said second rule set to at least in part govern at least one factor related to use of said portion of said first protected information.	When the second rule set is the conditional syntax statements written by the .NET assembly developer and placed into the signed .msi file, it governs the offer/installation of the .NET component. When the second rule set is the declarative statements in the .NET assembly's header,

	it governs the use of the .NET assembly, which includes the first protected information.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

85. A method comprising the following steps:	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
creating a first secure container comprising a first rule set and first protected information;	The first protected information is the .NET component. The first secure container is a signed .NET component (first protected information) governed by the set of declarative statements comprising the LicenseProviderAttribute (first rule set). The second alternative for the first secure container is a cabinet file signed by the .NET component developer containing a (signed or unsigned) .NET component with license support. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute.
storing said first secure container in a first memory;	The first secure container is stored at the .NET component developer's location.
creating a second secure container comprising a second rule set;	The first alternative for the second secure container is a signed .NET assembly and the second rule set is declarative statement(s) within the assembly's header. The second alternative for the second secure container is the signed .msi file in which the .NET assembly developer packages its (signed or unsigned) assembly. The second rule set is the conditional syntax statement(s) written by the .NET assembly developer and placed into the signed .msi file.
storing said second secure container in a second memory;	The second secure container is stored at the .NET assembly developer's location.
copying or transferring at least a first portion of said first protected information to said second secure container, said copying or transferring step comprising:	The .NET component developer packages its module in a signed .msi file for distribution to the .NET assembly developer's site.
creating a third secure container comprising a third rule set;	The third secure container is the signed .msi file in which the .NET component developer packaged its .NET component. The third control set is the conditional syntax statements written by the .NET component developer and placed into the signed .msi file.
copying said first portion of said	In preparation for using a msi authoring

first protected information;	tool, such as Microsoft's Orca, copying the .NET component to a package staging area.
transferring said copied first portion of said first protected information to said third secure container; and	Using the msi authoring tool to import the .NET component into the signed .msi file.
copying or transferring said copied first portion of said first protected information from said third secure container to said second secure container.	The .NET assembly developer installs the .NET component, which involves removing it from the .NET component developer's signed .msi file and installing it into its environment. Subsequently, the .NET assembly developer places the .NET component into its .NET assembly and/or signed .msi file when it is packaging its .NET assembly.
87. A method as in claim 85 in which said copied first portion of said first protected information consists of the entirety of said first protected information.	The entire .NET component is copied.
89. A method as in claim 85 in which	
said first memory is located at a first site,	The first memory is located at the .NET component developer's site.
said second memory is located at a second site remote from said first site, and	The second memory is located at the .NET assembly developer's site.
said step of copying or transferring said first portion of said first protected information to said second secure container further comprises copying or transferring said third secure container from said first site to said second site.	The .NET component developer's signed .msi file is transferred from its site to the site of the .NET assembly developer.
94. A method as in claim 85 further comprising:	
creating a fourth rule set.	When the second secure container is not a signed .NET assembly, the fourth rule set is declarative statements within the assembly's header. When the second secure container is not the signed .msi file in which the .NET assembly developer packages its (signed or unsigned) assembly, the fourth rule set is the conditional syntax statements written by the .NET assembly developer and placed into the signed .msi file.

293482.02

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

85 (alternate infringing scenario)	
A method comprising the following steps:	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
creating a first secure container comprising a first rule set and first protected information;	The first protected information is the .NET component. The first alternative for the first secure container is the signed and licensed .NET component. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component. The second alternative for the first secure container is a (signed or unsigned) .NET component with license support contained within a cabinet file signed by the .NET component developer. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component. The third alternative for the first secure container is the signed .msi file in which the .NET component developer packaged its assembly. The first rule set is the conditional syntax statements written by the .NET component developer and placed into the signed .msi file.
storing said first secure container in a first memory;	The first secure container is stored at the .NET component developer's location.
creating a second secure container comprising a second rule set;	The first alternative for the second secure container is a signed .NET assembly and the second rule set is declarative statement(s) within the assembly's header. The second alternative for the second secure container is the signed .msi file in which the .NET assembly developer packages its (signed or unsigned) assembly. The second rule set is the conditional syntax statement(s) written by the .NET assembly developer and placed into the signed .msi file.
storing said second secure container in a second memory;	The second secure container is stored at the .NET assembly developer's location.
copying or transferring at least a first	The .NET assembly developer places the

portion of said first protected information to said second secure container, said copying or transferring step comprising:	.NET component into the third secure container, which is either a signed cabinet file or a signed .NET assembly.
creating a third secure container comprising a third rule set;	When the second secure container is the signed .msi file, the third secure container is the signed .NET assembly. The third rule set is the declarative statement(s) in the .NET assembly's header. When the second secure container is either a .NET assembly or the signed .msi file, the third secure container is a signed cabinet file in which the .NET assembly developer placed licensed .NET component. The third rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component.
copying said first portion of said first protected information;	Copying the .NET component to either the .NET assembly or to the signed cabinet file.
transferring said copied first portion of said first protected information to said third secure container; and	Transferring the .NET component to either the .NET assembly or the signed cabinet file.
copying or transferring said copied first portion of said first protected information from said third secure container to said second secure container.	When the second secure container is the signed .msi file and the third secure container is the signed .NET assembly, the .NET assembly is placed into the signed .msi file. When the second secure container is either the .NET assembly or the signed .msi file and the third secure container is the signed cabinet file, the signed cabinet file is placed into either the .NET assembly or the signed .msi file.

87. A method as in claim 85 in which said copied first portion of said first protected information consists of the entirety of said first protected information.	The entire .NET component is copied.

93. A method as in claim 85 in which	
said step of copying transferring said copied first portion of said first protected information from said third secure container to said second secure container further comprises storing said third secure container in said second secure container.	When the third secure container is the signed .NET assembly, it is placed in the signed .msi file. When the third secure container is a signed cabinet file, it can be placed in either the .NET assembly and/or the signed .msi file.

94. A method as in claim 85 further comprising:	
creating a fourth rule set.	When the second rule set is declarative statement(s) within the assembly's header.

	the fourth rule set is the conditional syntax statement(s) written by the .NET assembly developer and placed into the signed .msi file. When the second rule set is the conditional syntax statement(s) written by the .NET assembly developer and placed into the signed .msi file, the fourth rule set is declarative statement(s) within the assembly's header or the set of declarative statements comprising the LicenseProviderAttribute in the .NET component.
95. A method as in claim 94 further comprising:	
using said fourth rule set to govern at least one aspect of use of said copied first portion of said first protected information.	If the fourth rule set is the .NET assembly developer's declarative statement(s) within the .NET assembly's header, it governs the use of the .NET component. If the fourth rule set is the conditional syntax statements of the .NET assembly developer's signed .msi file, it governs the offer/installation of the .NET component.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,915,019

85 (second alternate scenario for .NET)	Infringing products include the .NET Framework SDK, Microsoft Visual Studio .NET, the Microsoft Installer SDK, and products that include the Microsoft .NET CLR, and the Microsoft Installer technology.
A method comprising the following steps:	
creating a first secure container comprising a first rule set and first protected information;	The first protected information is a .NET component. The first alternative for the first secure container is the signed and licensed .NET component. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component. The second alternative for the first secure container is a (signed or unsigned) .NET component with license support contained within a cabinet file signed by the .NET assembly developer. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component. The third alternative for the first secure container is a .NET component whose hash is included in the assembly header of a .NET assembly. The first rule set is the set of declarative statements comprising the LicenseProviderAttribute in the .NET component.
storing said first secure container in a first memory;	The first secure container is stored at the .NET assembly developer's location.
creating a second secure container comprising a second rule set;	The second secure container is the signed .msi file in which the .NET assembly developer packages its signed assembly. The second rule set is the conditional syntax statement(s) written by the .NET assembly developer and placed into the signed .msi file.
storing said second secure container in a second memory;	The second secure container is stored at the .NET assembly developer's location.
copying or transferring at least a first portion of said first protected information to said second secure container, said copying or transferring step comprising:	The .NET assembly developer places the .NET component into the third secure container, which is the signed .NET assembly.
creating a third secure container comprising a third rule set;	The third secure container is a signed .NET assembly and the third rule set is

	declarative statement(s) within the assembly's header.
copying said first portion of said first protected information;	Copying the .NET component to the .NET assembly.
transferring said copied first portion of said first protected information to said third secure container; and	Transferring the .NET component to the .NET assembly.
copying or transferring said copied first portion of said first protected information from said third secure container to said second secure container.	When the second secure container is the signed .msi file and the third secure container is the signed .NET assembly, the .NET assembly is placed into the signed .msi file.

87. A method as in claim 85 in which said copied first portion of said first protected information consists of the entirety of said first protected information.	The entire .NET component is copied.

90. A method as in claim 85 in which	
said first memory and said second memory are located at the same site.	First and second memory is at the .NET assembly developer's location.

93. A method as in claim 85 in which	
said step of copying transferring said copied first portion of said first protected information from said third secure container to said second secure container further comprises storing said third secure container in said second secure container.	When the third secure container is the signed .NET assembly, it is placed in the signed .msi file.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,915,019

96. A method comprising performing the following steps within a virtual distribution environment comprising one or more electronic appliances and a first secure container, said first secure container comprising a first control set and first protected information:	A signed and licensed .NET component (first container) is part of a .NET assembly (second container), which is packaged in a signed .msi file (third container).
using at least one control from said first control set to govern at least one aspect of use of said first protected information while said first protected information is contained within said first secure container;	The first secure container is a licensed and signed .NET component governed by the set of declarative statements comprising the LicenseProviderAttribute (one control).
creating a second secure container comprising a second control set for governing at least one aspect of use of protected information contained within said second secure container;	The second secure container is a .NET assembly, the protected information is the assembly and the second control set is declarative statement(s) within the assembly's header.
incorporating a first portion of said first protected information in said second secure container, said first portion made up of some or all of said first protected information;	Included in the .NET assembly is the .NET component.
using at least one control to govern at least one aspect of use of said first portion of said first protected information while said first portion is contained within said second secure container; and	The declarative statement(s) govern the use of the .NET component and the custom LicenseProvider class (first control set) controls the .NET component.
incorporating said second secure container containing said first portion of said first protected information within a third secure container comprising a third control set.	The third secure container is the signed .msi file in which the .NET assembly developer packages its assembly. The third control set is the conditional syntax statements written by the assembly developer and placed into the signed .msi file.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,949,876

2.	Infringement is based on Microsoft's Visual Studio .NET and/or the .NET Framework licensing tools (in the.NET Framework SDK) and/or Microsoft Installer SDK..
A system for supporting electronic commerce including:	
means for creating a first secure control set at a first location;	The first location is a .NET component developer's site. The first secure control set is the set of declarative statements comprising the *LicenseProviderAttribute* of a first .NET licensed component that provides for a design-time license to use the control. This attribute also specifies the type of license validation that occurs. The component is encapsulated in a signed .NET assembly.
means for creating a second secure control set at a second location;	The second location is the .NET application developer's site where a .NET application comprising one or more assemblies is created. The second secure control set comprises the declarative statement(s) (including licensing statements, and code access security statements) of a signed .NET assembly using or calling the first .NET component. The control set can include a set of security permissions demanded by the .NET assembly containing the licensed component, whereby the permissions are demanded of components that call the application components. The control set can also be extended by controls expressed as conditional syntax statements in a signed .msi file containing a click through end-user license (the end-user license scenario).
means for securely communicating said first secure control set from said first location to said second location; and	The first .NET control set is securely communicated from the first location developer to the .NET solution provider by either being contained in a signed assembly, within a signed cabinet file or within a signed .msi file.
means at said second location for securely integrating said first and second control sets to produce at least a third control set comprising plural elements together comprising an electronic value chain extended agreement.	At the second location, the solution developer uses the .NET runtime that includes the LicenseManager. Whenever a class (control or component) is instantiated (here, an instance of the first .NET licensed component), the license manager accesses the proper validation mechanism for the control or component. A value chain is created through the creation of a run-time license for use of the first .NET component in the context of use of the .NET application developed at the second location. The

	license controls for the runtime license (derived from the design time license) are bound into the header of the .NET application assembly, along with the second control set. The creation of runtime license controls is securely handled by Visual Studio.NET or the LC tool. Runtime licenses are embedded into (and bound to) the executing assembly. The license control attribute included in the first .NET component is customized in the second location to express and require the runtime license. In a different scenario, the LC tool is used to create a ".licenses file" containing licenses for multiple components, including runtime licenses for components and classes created by the license provider. This .licenses file is embedded into the assembly. The third control set is an extended value chain agreement that comprises the runtime license controls for the first .NET licensed class (that had been bound to the assembly), the declarative controls provided by the solution provider in the solution provider's assembly, and any runtime licenses for other components included by the solution provider in the solution provider's assembly, and any end user license agreement provided by the application provider. The controls are typically integrated into the header of the .NET application assembly calling the first .NET licensed component. A further "end user licensing scenario" occurs when, at the second location, the application developer packages the application into a signed .msi file that includes conditional syntax statement controls that require that a user read and agree to an end user license agreement for the application and the embedded first component. The third control set includes a plurality of elements that include the run-time licenses mentioned above, security permissions controls, EULA controls (a fourth control set), all securely bound into the signed .msi file.

11. A system as in claim 2 in which said first location and said second location are contained within a Virtual Distribution Environment.	The Microsoft .NET Framework provides a Virtual Distribution Environment. Here the nodes are the Common Language Runtime instances that interpret the controls contained within .NET assemblies (among other functions).
29. A system as in claim 2 in which said first secure control set includes required	The licensing control in the first control set specifies the method required to validate

terms.	the license.
32. A system as in claim 2 in which said second secure control set includes required terms.	The security permissions demanded (as described above) are required terms for execution of the application code elements.
60. A system as in claim 2 in which said means for securely integrating said first and second control sets includes a fourth control set.	In the scenario where the application assembly is distributed using a signed .msi file, the secure integration of the first and second control sets is enhanced by the tamper protection afforded by the signed .msi file. In the end user license scenario, a fourth control set consisting of conditional syntax statements is included in the .msi file.
130. A system as in claim 2 further including means for executing said third control set within a protected processing environment.	The third control set is executed under the auspices of the CLR.
132. A system as in claim 130 in which said protected processing environment is located at a location other than said second location.	The third control set is executed at an end-user site within the CLR.
161. A system as in claim 2 in which said third control set includes controls containing human-language terms corresponding to at least certain of the machine-executable controls contained in said third control set.	In the end user license scenario, the third control set includes a fourth control set that requires that the human user agree with license terms displayed to the user. These human readable terms are referenced in the conditional syntax statement controls contained in the signed .msi file.
162. A method as in claim 161 in which said human-language terms are contained in one or more data descriptor data structures.	The .msi file is a data descriptor data structure.
170. A system as in claim 2 in which said means for creating a first secure control set includes a protected processing environment.	The creation of the first licensed component, including its licensed controls is carried out under the auspices of the CLR.
171. A system as in claim 2 in which said means for creating a second secure control set includes a protected processing environment.	The application design time environment and the creation of the .NET application is carried out under the auspices of the CLR.
172. A system as in claim 2 in which said means at said second location for securely integrating includes a protected processing environment.	The means for integrating the runtime license with the application controls is carried out under the auspices of the CLR.
329. A system as in claim 2 in which said	VS.NET runs under Windows.

PAGE 65/90 * RCVD AT 8/4/2004 8:45:44 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:6508496775 * DURATION (mm-ss):28-14

means for creating a first secure control set includes an operating system based on or compatible with Microsoft Windows.	
330. A system as in claim 2 in which said means for creating a second secure control set includes an operating system based on or compatible with Microsoft Windows.	VS.NET runs under Windows.
331. A system as in claim 2 in which said means at said second location for securely integrating said first and second control sets includes an operating system based on or compatible with Microsoft Windows.	VS.NET runs under Windows.
346. A system as in claim 2 further comprising means by which said third control set governs the execution of at least one load module.	The third control set in the scenario described in the claim map for claim 2 governs a portable .NET executable designed to be loaded into the CLR environment (a CLR host).
347. A system as in claim 2 further comprising means by which said third control set governs the execution of at least one method.	The third control set in the scenario described in the claim map for claim 2 governs a .NET executable. This executable contains one or more methods.
349. A system as in claim 2 further comprising means by which said third control set governs the execution of at least one procedure.	The third control set in the scenario described in the claim map for claim 2 governs a .NET executable. This executable contains one or more procedures.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,112,181

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
48.	Infringing products include Microsoft SMS (Systems Management Server) 2.0 and subsequent versions.
A method for narrowcasting selected digital information to specified recipients, including:	
a) at a receiving appliance, receiving selected digital information from a sending appliance remote from the receiving appliance,	The *receiving appliance* is the client (e.g., end user computer in an Enterprise setting) receiving *digital information* (packages and/or advertisement files) from the *sending appliance*, the centralized SMS database via a Client Access Point and/or Distribution Point set up on a server.
the receiving appliance having a secure node and being associated with a specified recipient;	The "node" is "secure" as a result of SMS security, as well as how it identifies and selects clients. The "specified recipient" is the result of the *collection* identifying a specific client that meets the criteria for a package or advertisement.
i) the digital information having been selected at least in part based on the digital information's membership in a first class, wherein the first class membership was determined at least in part using rights management information; and	The *digital information* is a software package or advertisement. The "*first class membership was determined in part using rights management information*" reads on creating software packages (or advertisements) based on attributes of the software.
ii) the specified recipient having been selected at least in part based on membership in a second class, wherein the second class membership was determined at least in part on the basis of information derived from the specified recipient's creation, use of, or interaction with rights management information; and	The "specified recipient" is the client selected to receive a package or advertisement. That recipient is chosen based on a collection rule, or on the recipient's possession of a license.
b) the specified recipient using the receiving appliance to access the received selected digital information in accordance with rules and controls, associated with the selected digital information,	The *receiving appliance* is the client computer. The SMS agents on the client computer receive, evaluate and take the appropriate action based on *rules and controls* governing the package and/or advertisement (i.e. the *selected digital information*).
the rules and controls being enforced	Rules and controls are enforced by Agents on

by the receiving appliance secure node.	the client (the *secure node*)
59. The method of claim 48 wherein said received selected digital information is at least in part event information.	Event information includes SMS event information, including *Scheduling Classes*.
63. The method of claim 48 wherein said received selected digital information is at least in part executable software.	All SMS packages must include a minimum of one program.
70. The method of claim 48 wherein said rules and controls at least in part govern usage audit record creation.	A control governs whether a MIF (management information file) is sent back to the SMS db after installation is done to report on the success or failure of the installation.
89. The method of claim 48 wherein said receiving appliance is a personal computer.	The primary purpose of SMS is to manage software on personal computers throughout the Enterprise.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,112,181

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
48.	Infringing products include Windows Media Player and Windows Media Rights Manager
A method for narrowcasting selected digital information to specified recipients, including:	This claim pertains to Windows Media Player with Individualized DRM Client and Windows Media Rights Manager used in the context of a narrowcast pay-per-view (hear) media distribution service., simulcast and/or subscription services.
(a) at a receiving appliance, receiving selected digital information from a sending appliance remote from the receiving appliance, the receiving appliance having a secure node and being associated with a specified recipient	Receiving appliance is a user's PC with individualized DRM client (secure node). Specified recipient is a user using the specific individualized DRM client to access and render narrowcast pay-per-view media, simulcast and/or subscription services for which the user acquires a license.
(i) the digital information having been selected at least in part based on the digital information's membership in a first class, wherein the first class membership was determined at least in part using rights management information; and	The digital information is media that is narrowcast to licensed recipients. These narrowcast streams are licensed to users who have acquired licenses and whose PCs (appliances) support WMPs that have individualized DRM clients. This attribute is included in the signed WMA file header and is used in the process of acquiring licenses for access to the media. Media that are licensed to the recipient have their licenses bound to the recipient's Individualization module.
(ii) the specified recipient having been selected at least in part based on membership in a second class, wherein the second class membership was determined at least in part on the basis of information derived from the specified recipient's creation, use of, or interaction with rights management information; and	The recipient is selected for this content based on the fact that the recipient is a member of the class of recipients who have a license for the narrowcast media and whose devices support WMP and individualized DRM clients. The recipient's machine must indicate support for individualization in challenges that are sent as part of requests for media in this narrowcast class.
(b) the specified recipient using the receiving appliance to access the received selected digital information in accordance with rules and controls, associated with the selected digital information, the rules and controls being enforced by the receiving appliance secure node.	Recipient's machine uses WMP and the individualized DRM client to access the narrowcast media in accordance with all rules associated with the media and contained in the media license – in particular, requirements that individualization be supported.

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
61. The method of claim 48 wherein said received selected digital information is at least in part entertainment information.	The digital information is Windows Media, which encodes audio/visual entertainment content.
62. The method of claim 61 wherein said entertainment information is at least in part music information.	Reads on narrowcast Windows Media Files that are music or audio/visual.
67. The method of claim 48 wherein said rules and controls at least in part use digital certificate information.	The license contains a digital certificate. The DRM client uses the certificate in the license to verify this signature and to verify that the header has not been tampered with.
72. The method of claim 48 wherein said rules and controls in part specifying at least one clearinghouse acceptable to rightsholders.	The signed header contains at least one URL that indicates to the Windows Media Rights Manager the license clearinghouse to be used in acquiring licenses.
75. The method of claim 72 wherein said at least one acceptable clearinghouse is a rights and permissions clearinghouse.	This clearinghouse is a license clearinghouse responsible for mapping rights and permissions onto requested content or narrowcasts and binding them to the requesting client environment or user of this environment.
89. The method of claim 48 wherein said receiving appliance is a personal computer.	Windows Media Player and the Individualized DRM client run on a personal computer.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.

INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,112,181

91	Infringing products include Windows Media Player and Windows Media Rights Manager
A method for securely narrowcasting selected digital information to specified recipients including:	This claim pertains to Windows Media Player with Individualized DRM Client and Windows Media Rights Manager used in the context of a narrowcast simulcast, pay-per-view (hear) media distribution service. and/or subscription services. The content is delivered in a Protected Windows Media File.
(a) receiving selected digital information in a secure container at a receiving appliance remote from a sending appliance, the receiving appliance having a secure node, the receiving appliance being associated with a receiving entity	Narrowcast content is received in a Protected Windows Media File. Receiving appliance is user's PC with individualized DRM client (secure node).
(i) the digital information having been selected at least in part based on the digital information's membership in a first class,	The digital information is media that is narrowcast to licensed recipients (for example, a sold-out concert is narrowcast on the Internet to "the class of" licensed (or ticketed) viewers).
(ii) the first class membership having been determined at least in part using rights management information	These narrowcast streams are licensed to users who have acquired licenses and whose PCs (appliances) support WMPs that have individualized DRM clients. This attribute is included in the signed WMA file header and is used in the process of acquiring licenses for access to the media. Media that are licensed to the recipient have their licenses bound to the recipient's individualization module.
(b) the receiving entity having been selected at least in part based on said receiving entity's membership in a second class,	The recipient is selected for this content based on the fact that the recipient is a member of the class of recipients who has a license for the narrowcast media.
(i) the second class membership having been determined at least in part on the basis of information derived from the recipient entity's creation, use of, or interaction with rights management information	The recipient class is determined by the license bound to the user's device that supports WMP and individualized DRM clients. The recipient's machine must indicate support for individualization in challenges that are sent as part of requests for media in this narrowcast class.
(c) receiving at the receiving appliance rules and controls in a secure container,	Receives a protected Windows Media File
(i) the rules and controls having been associated with the selected digital information; and	Receives a license that is bound to the file as well as to the specific DRM client individualization information.
(d) using at the receiving appliance the selected digital information in accordance	Recipient's machine uses WMP and the individualized DRM client to access the

Exhibit B

293482.02

with the rules and controls,	narrowcast media in accordance with all rules associated with the media and contained in the media license – in particular, requirements that individualization be supported.
(i) the rules and controls being enforced by the receiving appliance secure node.	The WMP and DRM client enforce the rules embedded in the Protected Windows Media File License.
104. The method of claim 91 wherein said received selected digital information includes entertainment information.	The digital information is Windows Media, which encodes audio/visual entertainment content.
109. The method of claim 91 wherein said rules and controls at least in part use digital certificate information.	The license contains a digital certificate. The DRM client uses the certificate in the license to verify this signature and to verify that the header has not been tampered with.
114. The method of claim 91 wherein said rules and controls specify at least one clearinghouse acceptable to rightsholders.	The signed header contains at least one URL that indicates to the Windows Media Rights Manager the license clearinghouse to be used in acquiring licenses.
117. The method of claim 114 wherein said at least one acceptable clearinghouse is a rights and permissions clearinghouse.	This clearinghouse is a license clearinghouse responsible for mapping rights and permissions onto requested content or narrowcasts and binding them to the requesting client environment or user of this environment.
131. The method of claim 91 wherein said receiving appliance is a personal computer.	Windows Media Player and the individualized DRM client run on a personal computer.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,389,402

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
1.	Products infringing: Microsoft Visual Studio .NET, .NET License Compiler, .NET Framework SDK, and .NET Common Language Runtime
A method including	A method for producing a third .NET component (application) that incorporates first and second .NET component whose distribution is license controlled.
creating a first secure container including a first governed item and having associated a first control;	The *first secure container* is a first signed .NET component that includes a license control. The *governed item* is the .NET component. The *first control* is the set of declarative statements comprising the LicenseProviderAttribute of a first .NET licensed component that provides for a design-time license to use the control. This attribute also specifies the type of license validation that occurs.
creating a second secure container including a second governed item and having associated a second control;	The *second secure container* is the second signed .NET component that includes a license control. The *governed item* is the .NET component. The *second control* is the set of declarative statements comprising the LicenseProviderAttribute of a second .NET licensed component that provides for a design-time license to use the control. This attribute also specifies the type of license validation that occurs.
transferring the first secure container from a first location to a second location;	The creator distributes a signed and licensed .NET component. An application developer at a second location downloads a first .NET component for inclusion into an application.
transferring the second secure container from a third location to the second location;	A creator distributes a signed and licensed .NET component from a different location. Application developer downloads a second .NET component for inclusion into an application.

at the second location, obtaining access to at least a portion of the first governed item, the access being governed at least in part by the first control;	At the *second location*, the application developer uses the .NET runtime that includes the LicenseManager to access a *first governed item*. Whenever a class (control or component) is instantiated (here, an instance of the first .NET licensed component), the license manager accesses the proper validation mechanism for the control or component. The *first control* comprises the declarative statement(s) (including licensing statements, and code access security statements) of the first .NET component.
at the second location, obtaining access to at least a portion of the second governed item, the access being governed at least in part by the second control;	At the *second location*, the application developer uses the .NET runtime that includes the LicenseManager to access a *second governed item*. Whenever a class (control or component) is instantiated (here, an instance of the second .NET licensed component), the license manager accesses the proper validation mechanism for the control or component. The *second control* comprises the declarative statement(s) (including licensing statements, and code access security statements) of the second .NET component.
at the second location, creating a third secure container including at least a portion of the first governed item and at least a portion of the second governed item and having associated at least one control, the creation being governed at least in part by the first control and the second control.	At the *second location*, the application developer uses the .NET runtime that includes the LicenseManager to access a *first governed item* and *second governed item* to construct an application, the *third secure container*. *Creation governance* is accomplished by invoking the .NET runtime to access the *first governed item* and the *second governed item*. Whenever a class (control or component) is instantiated the license manager accesses the proper validation mechanism for the control or component. The *portions* of the first governed item and second governed item that are being included in the third secure container will typically include the governed items themselves, ie. the .NET components. The *associated control* in this case is the LicenseProviderAttribute, created and inserted into the application.

EXHIBIT C

Exhibit C

EXHIBIT C

CONFIDENTIAL—SUBJECT TO PROTECTIVE ORDER OF NOVEMBER 19, 2001: Exhibit C contains documents or things that are the subject of a Protective Order of this Court and cannot be opened or its contents made available to anyone other than this Court or counsel of record for the parties.

Exhibit 2

WILLIAM L. ANTHONY (State Bar No. 106908)
ERIC L. WESENBERG (State Bar No. 139696)
HEIDI L. KEEFE (State Bar No. 178960)
BAS DE BLANK (State Bar No. 191487)
ORRICK, HERRINGTON & SUTCLIFFE, LLP
1000 Marsh Road
Menlo Park, CA 94025
Telephone: (650) 614-7400
Facsimile: (650) 614-7401

STEVEN ALEXANDER (admitted *Pro Hac Vice*)
JAMES E. GERINGER (admitted *Pro Hac Vice*)
JOHN D. VANDENBERG
KLARQUIST SPARKMAN, LLP
One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, OR 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446

Attorneys for Defendant and Counterclaimant,
MICROSOFT CORPORATION

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

OAKLAND DIVISION

INTERTRUST TECHNOLOGIES CORPORATION, a Delaware corporation, Plaintiff, v. MICROSOFT CORPORATION, a Washington corporation, Defendant.	Case No. C 01-1640 SBA (MEJ) Consolidated with C 02-0647 SBA (MEJ) **DEFENDANT MICROSOFT CORPORATION'S PRELIMINARY INVALIDITY CONTENTIONS** **(Patent Local Rules 3-3 and 3-4)**
AND RELATED CROSS-ACTION.	

I. Patent Local Rule 3-3(a) Identification of Prior Art

Pursuant to Patent Local Rule 3-3, Defendant Microsoft Corporation ("Microsoft") makes the following Preliminary Invalidity Contentions[1] with respect to the following patents asserted by plaintiff InterTrust Technologies Corporation ("InterTrust") in this action: U.S. Patent No. 6,185,683 ("the `683 patent"); U.S. Patent No. 6,253,193 ("the `193 patent"); U.S. Patent No. 5,920,861 ("the `861 patent"); U.S. Patent No. 5,982,891 ("the `891 patent"); U.S. Patent No. 5,917,912 ("the `912 patent"); U.S. Patent No. 6,157,721 ("the `721 patent"); U.S. Patent No. 5,915,019 ("the `019 patent"); U.S. Patent No. 5,949,876 ("the `876 patent"); U.S. Patent No. 6,112,181 ("the `181 patent"); and U.S. Patent No. 6,389,402 ("the `402 patent").

Despite the length of time this case has been pending, discovery is still at an early stage due to intervening stays. InterTrust continues to assert eleven patents and over one hundred and fifty claims. In view of these factors, Microsoft continues to evaluate the prior art at this time. Microsoft reserves the right to amend or supplement its Preliminary Invalidity Contentions to take into account prior art, information or defenses that might come to light as a result of its continuing discovery efforts, errors subsequently recognized by any of the parties, and as a result of further evaluation of the prior art.[2] In addition, Microsoft has moved to strike InterTrust's September 2, 2003 PLR 3-1 Preliminary Infringement Contentions as being insufficient. To the extent that the Court grants Microsoft's motion and orders InterTrust to amend/re-serve its 3-1 statement in compliance with the Local Rules, Microsoft reserves the right to amend or supplement its PLR 3-3 Preliminary Invalidity Contentions in response to any amended infringement contentions submitted by InterTrust. Microsoft further reserves the right to rely

[1] These Preliminary Invalidity Contentions incorporate by reference Microsoft's prior Preliminary Invalidity Contentions dated August 7 and 16, 2002.

[2] For example, Microsoft reserves the right to amend/supplement this disclosure once InterTrust complies with discovery responses, which Microsoft contends are incomplete and inadequate. To date, Microsoft has objected to InterTrust's continued refusal to provide information sought in discovery, including, but not limited to: the identity of the alleged inventors of specific claims; conception or actual reduction to practice dates for specific claims; whether to there has ever been any alleged embodiment(s) of the asserted claims; and what, if any, specification support is alleged, including from any of the applications for which InterTrust claims priority. Each of these pieces of information could affect the priority date for any given claim, expanding or narrowing the window of applicable prior art. Without this information, which is within InterTrust's exclusive knowledge and control, Microsoft's PLR 3-3 Contentions are subject to amendment and/or supplementation.

upon InterTrust's own activities, alone and in connection with others. Microsoft further reserves the right to amend this statement or otherwise further respond if InterTrust contends (or the Court rules) that any earlier or later priority dates may apply for individual claims. Microsoft also reserves its right to amend or supplement these invalidity contentions pursuant to Patent Local Rule 3-6 and 3-7.

Attached hereto, as Appendix A, is a listing showing "the identity of each item of prior art that allegedly anticipates each asserted claim or renders it obvious" (PLR 3-3(a)). On information and belief, each listed publication became prior art at least as early as the dates given. In addition, the citations and explanations provided in the exhibits are mere examples, and Microsoft reserves its right to rely on any other portions or aspects of the prior art references and systems that may also disclose or practice elements of the asserted claims. Patent Local Rule 3-3 does not require identification of evidence that establishes the inherence of a claim element in an item of prior art, nor does it require identification of evidence that establishes knowledge of those of ordinary skill in the relevant fields of art. Accordingly, Microsoft does not purport to have provided all such information in the attached exhibits.

From InterTrust's current document production, it appears that its employees' and consultants' activities, including offers for sale, public uses, derivations, "inventions" (as the word is used in 35 U.S.C. § 102(g)), and disclosures to Willis Ware, Drew Dean, and others not under any duty of confidentiality, constituted or created material and perhaps anticipatory prior art to many of the asserted claims. This art was not cited to the Patent Office. Discovery is ongoing, and Microsoft reserves the right to amend or supplement this disclosure after Microsoft has had an opportunity to investigate this possible prior art during discovery.

II. Patent Local Rule 3-3(b) and 3-3 (c) Classification and Analysis of Prior Art

Microsoft contends that at least one term or phrase in each of the asserted claims is indefinite under 35 U.S.C. § 112, and hence, each of the asserted claims is incapable of construction. However, for the limited purpose of classification and analysis of prior art, Microsoft has construed the claim terms in a manner consistent with the apparent construction of terms offered by InterTrust in its Revised Preliminary Infringement Contentions. <u>Microsoft does</u>

not agree with these constructions, and nothing in these Preliminary Invalidity Contentions should be construed as an admission, a declaration against interest, whether under the Federal Rules of Evidence or otherwise, as to what a particular claim limitation means. For this reason, Microsoft's identification of "corresponding structures" for "means-plus-function" limitations that are set out in the Preliminary Invalidity Charts are not admissions as to the identity of such structures. Rather, they are based upon Microsoft's best guess as to what InterTrust may someday identify as corresponding structures for the means-plus-function limitations of its asserted claims, to the extent that Microsoft understands them.[3]

Accordingly, Microsoft's Preliminary Invalidity Contentions should not be construed as advocating a particular claim construction for any disputed claim terms. For the limited purpose of providing Preliminary Invalidity Contentions, and subject to the conditions set forth above, Microsoft has, to the extent possible, attempted to construe the claims in a manner consistent with InterTrust's Revised Preliminary Infringement Contentions.

Pursuant to Patent Local Rules 3-3(b) and 3-3(c), Microsoft provides the classification of prior art in the listing and charts attached hereto as Appendices A and B. Appendix A, beyond identifying each item of prior art, further indicates whether each prior art reference is used as an anticipatory reference and/or as a reference which, alone, or in combination with other prior art, renders the claims obvious. Appendix B includes charts which (1) specifically identify where in each item of prior art each element of each asserted claim is found and (2) establish how that prior art anticipates or renders obvious all of the asserted claims. In the event that any charted prior art is found not to be anticipatory under 35 U.S.C. § 102, Microsoft reserves the right to rely upon that art to prove obviousness under 35 U.S.C. § 103. Likewise, in the event InterTrust

[3] To date, InterTrust has refused to identify any structure corresponding to the means-plus-function elements in its asserted claims. It is Microsoft's position that this is a violation of the Patent Local Rules, and that as a result of refusing to identify a structure associated with each means-plus-function element, InterTrust admits that there is no such structure disclosed, has waived its right to assert claimed structure, and that those claims are therefore invalid at least for failure to satisfy the written description requirement of 35 U.S.C. §112. *See* InterTrust's Patent Local Rule 3-1 served September 2, 2003 and InterTrust's Opposition to Microsoft's Motion to Strike InterTrust's PLR 3-1 Contentions.

amends or supplements its Preliminary Infringement Contentions, Microsoft reserves its rights to amend and supplement its Preliminary Invalidity Contentions.

To the extent that any prior art produced to InterTrust has not been classified as prior art under 35 U.S.C. §§ 102 or 103, Microsoft reserves the right to rely on this art or supplement its disclosure for the following reasons:

(i) Microsoft's position on the invalidity of particular claims will depend on how those claims are construed by the Court. As thus far only preliminary claim construction has occurred Microsoft cannot take a final position for the bases for invalidity of disputed claims. The Court's subsequent claim constructions of remaining terms may yield constructions different from what Microsoft assumes herein.

(ii) Microsoft is continuing to diligently search for relevant prior art but has not yet completed that search and continues to evaluate prior art that has been located.

(iii) Microsoft has not completed its discovery from Plaintiff or from third parties with knowledge of the relevant prior art. Depositions of the persons involved in the drafting and prosecution of the patents-in-suit, the inventors, and persons who attempted to practice InterTrust's claimed invention, for example, will likely affect Microsoft's contentions.

A. Prior Art Under 35 U.S.C. § 102 Which Anticipates The Asserted Claims of Each of the Asserted Patents

Subject to the above-referenced qualifications concerning the preliminary nature of this disclosure, Microsoft believes a reasonable basis exists that, as more particularly explained in the Preliminary Invalidity Contentions attached as Appendix B hereto, the references listed in Appendix B anticipate the asserted claims of the each of the asserted patents.

B. Prior Art Under 35 U.S.C. § 103 Which Renders Obvious One or More of the Asserted Claims

Each of the references called out in Appendix A can be combined with one another so as to render one or more of the claims of the asserted patents invalid as obvious, and many of them are explicitly motivated to do so by virtue of extensive cross-references to one another's solutions. InterTrust is currently asserting 151 claims in eleven patents, which cite hundreds of references. Hundreds of additional non-cited relevant prior art has been uncovered and cited to

InterTrust. The number of potential combinations of these references, if only two or a few references are combined for each claim, is necessarily very large. Microsoft requests InterTrust to reduce its asserted claims so as to reduce the number of combinations to a manageable number. Nonetheless, Microsoft has provided mapping of combinations as discussed below. Indeed, even where explicit cross-referencing and incorporation by reference does not exist, the motivation to combine any of the references arises from the common objectives and subject matter, digital rights management. The common objectives and subject matter are expressed generally in the claim charts of Appendix B, which are incorporated by reference into Microsoft's showing under 35 U.S.C. § 103.

The motivation for seeking "security," privacy and integrity was widely recognized in the United States and elsewhere prior to February 13, 1994, and since prior to February 13, 1994, has extended to any information or item of perceived value, including books, music, games, computer systems, other computer programs, and any digital data or content that maybe deemed valuable or worthy of protection. Additional motivations to combine references include the desire to meet or exceed any applicable laws or industry or government standards, such as the Orange Book, Computer Fraud and Abuse Act of 1986, Computer Security Act of 1989 PL100-35, High Performance Computing Act (HPCA) of 1991 (PL102-194), and 17 U.S.C. §§ 101 et seq. Industry standards include those for communication such as X.509, TCP/IP, WWW, and WAIS, and those for encryption or transmission of encrypted information, *e.g.* DES, Triple DES, RSA, SSL, MIME, S/MIME, SHTTP, HTTPS, MD5, and PEM. Additional teachings to combine these references with such items of information include "security" (including "security" levels), permissions, certificates, tickets, "secure" processors, "secure" storage, "smart" cards (including smart cards able to store data and perform computations such as encryption/decryption), tamper resistance techniques for hardware and software, physical "security", and "trusted" time. Also included are authentication and authorization in trusted distributed systems, enabling software or features thereof to run only on particular machines or in particular ways, and treating binary information/data at varied levels of granularity.

It was further obvious to combine any of these "security" features with any of the software or hardware available at the time. For example, it would have been obvious to combine any file and operating systems such as NT, NFS, Andrew, Netware, Mach, DT Mach, Multics, Amoeba, ISOS, and Unix; or protocols, codes and systems such as secure kernels, WWW, SSL, SGML, hyptertext, Oak, Telescript, OOP and other programming technologies or frameworks (*e.g.* Smalltalk, COM, OLE, Bento, OpenDoc; object oriented databases with watermarking; obfuscation; swIPe; SNMP; auditing; on-line (or other digitally transmitted) transaction and subscription-based services and billings; electronic payment; on-line banking, entertainment and commercial interactive commerce; ATMs; encryption and authentication; physical security tools and devices; physically secure locations; physically "secure" products such as tamper resistant computer or other devices, "secure" processors, "secure" memory, "smart" cards, set-top boxes, portable devices, "secure" communications facilities, electronic wallets.[4]

III. Patent Local Rule 3-3(d) Disclosure: Invalidity For Failure to Satisfy 35 U.S.C. § 112.

Each of the asserted InterTrust patent claims is invalid as indefinite, for inadequate written description and for lack of enablement as those requirement are set forth by 35 U.S.C. § 112.[5] In accordance with Patent L.R. 3-3(d), Microsoft identifies in Appendix C, attached hereto, exemplary bases, on an element by element basis, for invalidating each asserted claim of each asserted patent for indefiniteness and lack of an adequate written description. The asserted claims are unclear in scope and not nearly as precise as the subject matter allows.

Appendix C contains examples of why the indefiniteness prohibited by 35 U.S.C. § 112(2) arises from many causes, including:

a) use of terms that lack an ordinary meaning in the art and are undefined in the specification;

[4] These examples are not intended to be an exhaustive list and are set forth for illustrative purposes.
[5] Microsoft also asserts that one or more of the claims are invalid under 35 U.S.C. § 112(1) for failure to identify the "best mode" for carrying out the invention. However, pursuant to Patent L.R. 3-3(d), Microsoft's arguments related to that defense are not required to be set forth in the attached charts, and hence are not included in Exhibit C.

b) use of terms that are used in the specification in a manner which is internally inconsistent, as well as inconsistent with their ordinary meaning, but are not specifically defined in the specification;

c) InterTrust's refusal to identify the structure in the application's written description linked to claim elements subject to 35 U.S.C. § 112, ¶6 ("means (or step) plus function);

d) such excessive disclaimers of specificity of a term that the term becomes meaningless;

e) inconsistent uses of a term within a single specification;

f) inconsistent uses of a term between a specification and something allegedly incorporated into that specification;

g) inconsistencies within the language of a given claim;

h) inclusion of the same element twice in a claim, resulting in improper double inclusion of an element;

i) impermissible reference to trademarks in a claim;

j) inconsistent use of terms that may be synonyms for one another or that could be used to mean same thing or different things.

The indefiniteness of the asserted claims is exacerbated by InterTrust's attempt to apply these claims to the very different structures and techniques of (or those that InterTrust wrongly attributes to) the Microsoft accused products. Microsoft reserves the right to modify this listing, *e.g.*, if and when InterTrust clarifies its infringement contentions and claim construction positions.

Appendix C also provides examples of the lack of an adequate written description supporting the asserted claims. For example, the asserted claims fail for lack of an adequate written description under 35 U.S.C. § 112(1) to the extent that they are construed to contradict and/or fail to require the essential, non-optional alleged attributes of the alleged "inventions" identified in their specifications (and any specification allegedly incorporated by reference) and the applications from which the patents issued. The asserted claims also fail to comply with the written description requirement as set forth in *Gentry Gallery, Inc v. Berkline Corp.*, 134 F.3d 1473 (Fed. Cir 1998) to the extent that the scope of any of them exceeds the scope of the alleged "invention" as set forth in the accompanying specification (and any specification allegedly incorporated therein). For example, in the specification of U.S. Patent No. 6,253,193 InterTrust states that:

> The present invention assertedly provides a new kind of "virtual distribution environment" (called "VDE" in this document) that secures, administers, and audits electronic information use. VDE also features fundamentally important capabilities for managing content that travels "across" the "information highway." These capabilities comprise a rights protection solution that serves all electronic community members. These members include content creators and distributors, financial service providers, end-users, and others. VDE is the first general purpose, configurable, transaction control/rights protection solution for users of computers, other electronic appliances, networks, and the information highway.

Accordingly any claims that rely on this specification must be limited in scope to the invention described therein. To the extent that they exceed the scope of what is described, they are invalid under the written description requirement.

Microsoft further contends that each asserted claim, when viewed in its entirety, is invalid under 35 U.S.C. § 112(1) because the specifications of the patents fail to teach one of ordinary skill in the art how to practice the entirety of the broad scope of those claims without undue experimentation.

For example, based on the specification, most if not all of the claims involve the use of software of one kind or another, yet the specification does not disclose any software programs that could be used or adapted for use in practicing the claimed inventions. In addition to failing to disclose any software program by explicit reference, the patent specifications does not describe with sufficient specificity the identity of software programs needed to practice the claimed invention that would prevent the need for undue experimentation by a person skilled in the art to practice the claimed inventions. The claims set forth a multiplicity of functions, features, and characteristics for the purported inventions, and the specifications are replete with references to software necessary to practicing the inventions, yet the specification neither identifies enabling software that satisfies such requirements, nor provides guidance that would allow a person of ordinary skill in the art to program enabling software without undue experimentation.[6]

As shown in Appendix C[7], asserted claims contain terms that are subject to multiple definitions, and the patent specifications do not disclose one or more of the alternate definitions. The full scope of the claim is therefore not described or taught in the specification. Any claim in Appendix C that contains a claim term subject to multiple definitions fails to teach the full scope of the claim and therefore fails the enablement requirement if the specification does not specify the operative definition for the term.

There are numerous other reasons that the unprecedented breadth of scope of the claims asserted by InterTrust are not enabled, including InterTrust's failure to implement the claims after substantial investment of time, labor, and money. Given the complexity of the asserted patents and their interdisciplinary subject matter, the state of the prior art, the absence of predictability of the prior art, the amount of experimentation necessary to practice the patents, the absence of embodiments, and the absence of guidance for practicing the invention provided in the specification[8], the relative skill of those practicing the art and the breadth of the claims, the asserted claims fail to meet the enablement requirement of 35 U.S.C. § 112 ¶ 1.

The full claims of the asserted patents fail to satisfy the enablement and written description requirements for the following reasons:

The '683 Patent

Claim 2: Claim 2 of the '683 patent fails the enablement requirement because the specification does not teach a person of ordinary skill in the relevant arts how to practice the purportedly disclosed invention without undue experimentation in the development of enabling

[6] In its discovery responses, InterTrust refuses to identify software programs necessary for practicing the inventions purportedly disclosed in the asserted patents. *See* InterTrust responses to Microsoft Interrogatory Nos. 39 and 40.
[7] *See* Appendix C for further element by element analysis of invalidity for failure to satisfy 35 U.S.C. § 112 ¶ 1. The indefiniteness of the claim terms addressed in Exhibit C affect enablement because the indefiniteness of the claim terms prevents the specification from adequately teaching a person of skill in the art how to make and use the full scope of the claimed inventions without undue experimentation.
[8] The failure of the specifications to provide necessary guidance also establishes that the claims fail to meet the written description requirement of 35 U.S.C. § 112 ¶ 1.

software and operation of such software on accompanying hardware. Specifically, limitations in Claim 2 (63:40-66), both explicitly and implicitly require software. Since no software is disclosed in the specification, and since the specification provides no useful programming guidance, a person of skill in the art would have to engage a process of trial and error, perhaps followed by bottom up software development, in order to make and use the full scope of Claim 2. Claim 2 also fails the enablement requirement in light of the breadth of the subject matter claimed (*e.g.* "security", "secure container," "containing"). The specification does not teach a person of ordinary skill in the art how to practice the full scope of the claim, and a person of skill in the art would therefore be required to undertake undue experimentation in order to make and use the invention across the full scope claimed. For these reasons and for the reasons stated above with respect to all of the claims, Claim 2 fails the enablement and written description requirements of 35 U.S.C. § 112 ¶ 1.

Claim 3: Claim 3 of the '683 patent fails the enablement requirement because the specification does not teach a person of ordinary skill in the relevant arts how to practice the purportedly disclosed invention without undue experimentation in the development of enabling software and operation of such software on accompanying hardware. Specifically, several limitations in Claim 3 (64:6-30), both explicitly and implicitly require software. Since no software is disclosed in the specification, and insufficient programming guidance (if any) is provided by the specification, a person of skill in the art would have to engage a process of trial and error, perhaps followed by bottom up software development, in order to make and use the full scope of Claim 3. Claim 3 also fails the enablement requirement in light of the breadth of the subject matter claimed (*e.g.* "security", "secure container," "rule"). The specification does not teach a person of ordinary skill in the art how to practice the full scope of the claim, and a person of skill in the art would therefore be required to undertake undue experimentation in order to make and use the invention across the full scope claimed. For these reasons and for the reasons stated above with respect to all of the claims, Claim 3 fails the enablement and written description requirements of 35 U.S.C. § 112 ¶ 1.

Claim 4: Claim 4 is dependent upon Claim 3 and thus fails the enablement and written description requirements of 35 U.S.C. § 112 ¶ 1 for the reasons stated above. In addition, the limitation of Claim 4 fails because it requires additional undisclosed software.

Claim 5: Claim 5 of the '683 patent fails the enablement requirement because the specification does not teach a person of ordinary skill in the relevant arts how to practice the purportedly disclosed invention without undue experimentation in the development of enabling software and operation of such software on accompanying hardware. Specifically, several limitations in Claim 5 (64:41-66), both explicitly and implicitly require software. Since no software is disclosed in the specification, and no meaningful programming guidance is provided, a person of skill in the art would have to engage a process of trial and error, perhaps followed by bottom up software development, in order to make and use the full scope of Claim 5. Claim 5 also fails the enablement requirement in light of the breadth of the subject matter claimed (*e.g.* "security", "secure container," "governed item"). The specification does not teach a person of ordinary skill in the art how to practice the full scope of the claim, and a person of skill in the art would therefore be required to undertake undue experimentation in order to make and use the invention across the full scope claimed. For these reasons and for the reasons stated above with respect to all of the claims, Claim 5 fails the enablement and written description requirements of 35 U.S.C. § 112 ¶ 1.

Claim 6: Claim 6 is dependent upon Claim 5 and thus fails the enablement and written description requirements of 35 U.S.C. § 112 ¶ 1 for the reasons stated above. In addition, the limitation of Claim 6 fails because it requires additional undisclosed software..

Claim 28: Claim 28 of the '683 patent fails the enablement requirement because the specification does not teach a person of ordinary skill in the relevant arts how to practice the purportedly disclosed invention without undue experimentation in the development of enabling software and operation of such software on accompanying hardware. Specifically, several limitations in Claim 28 (70:20-59), both explicitly and implicitly require software. Since no software is disclosed in the specification, and no meaningful programming guidance is provided, a person of skill in the art would have to engage a process of trial and error, perhaps followed by bottom up software development, in order to make and use the full scope of Claim 28. Claim 28